Characterizing Peer-level Performance of BitTorrent

DRP Report

Amir H. Rasti

ABSTRACT

BitTorrent is one of the most popular Peer-to-Peer (P2P) content distribution applications over the Internet that significantly contributes in network traffic.

In BitTorrent, a file is divided into segments and participating peers contribute their outgoing bandwidth by providing their available segments to other peers while obtaining their missing peers from others. Characterization of BitTorrent is useful in determining its performance bottlenecks as well as its impact on the network.

In this study, we try to address the following two key questions through measurement: (i) *What are the main factors that affect observed performance by individual peers in BitTorrent?*, and (ii) *What are the contributions of these factors on the performance of individual peers?* To address these questions, first we examine the group-level and peer-level characteristics of BitTorrent using three tracker logs from different sources. Second, we use statistical analysis (namely rank correlation and linear regression) to determine and quantify the potential effects of both peer-level and group-level properties on the performance of individual peers.

We conclude that: (i) There is no single property that has dominant effect on the observed performance by individual peers, (ii) Outgoing bandwidth of each peer, average available content in the group and churn rate appear to have the most notable effect on peer performance and (iii) The behavior of the system in practice is rather complex due to the inherent dynamics in peer participation and content delivery as well as bandwidth heterogeneity and asymmetry.

1. INTRODUCTION

During the past few years, peer-to-peer applications have become very popular on the Internet. BitTorrent in one of the most popular peer-to-peer applications providing scalable peer-to-peer content distribution over the Internet. Some recent studies [8] have shown that BitTorrent is accountable for approximately 35% of the Internet traffic. BitTorrent is a scalable peer-to-peer content distribution system that enables one-to-many distribution of large files without requiring a large access link bandwidth at the source. Similar to other peer-to-peer systems, it uses resources of participating peers to increase the capacity of the system. The main shared resource in BitTorrent is the up-link bandwidth of individual peers. The file being distributed is divided into a large number of segments. The source provides different segments of the content to different peers. Participating peers connect to each other to form an overlay and peers exchange the available segments until they have downloaded the entire content.

This approach enables participating peers to contribute their outgoing bandwidth which makes BitTorrent a scalable system. Peers may stay in the system once their downloading is complete acting as a seed. This behavior whether intentionally or not, makes the available resources in the system richer and improves its robustness to system dynamics.

BitTorrent is widely used for the distribution of free software such as Linux as well as copies of the latest movies and music albums.

It is believed to successfully utilize participating peers' resources quickly and to be able to handle flash crowds with acceptable performance [9](i.e., download time) despite the dynamics of peer participation. It is also believed that BitTorrent's "tit-for-tat" is a successful incentive mechanism singling out free riders.

The goal of individual peer is to maximize their download rates by finding high speed uploaders and keeping them satisfied by providing them with a good upload rate. However, the extent to which each peer is successful in achieving this goal or its observed performance is the subject of this project. We believe that the performance a peer will observe, potentially depends on two sets of parameters: *group properties* and *peer properties*. The first set of parameters capture the group status from different aspects and are defined as a function of time: (i) group population, (ii) peer arrival and departure rates (churn), (iii) average content availability among participating peers and (iv) percentage of seeds are the main group properties that might affect a peer's performance. The second set are the properties of the peer itself regardless of the group. The peer's incoming and outgoing bandwidth, geographical location, local configuration, ratio of contribution (upload over download) are examples of peer properties that might potentially affect peer's performance. To capture peer properties, we record a signature of each peer's properties during its life time using statistical parameters.

While BitTorrent enables a single peer to distribute content to a large number of interested receivers, it is unclear what factor(s), determine the observed performance, namely download rate, by individual peers.

In this paper, our goal is to empirically answer the following basic questions about observed performance by individual peers in BitTorrent:

(i) What are the dominant key peer or group properties that are more likely to determine the observed performance by individual peers? (ii) What is the contribution of each property and how we can quantify that?

To answer these questions, we examine the observed performance by participating peers in three popular torrents using BitTorrent tracker logs. We identify different challenges in (i) estimating peer and group properties from BitTorrent tracker logs, (ii) dealing with large data set and (iii) identifying and coping with various errors and bugs in a data set. We also explain why assessing the observed performance by individual peers is not trivial.

Using the derived peer and group properties for three data sets, we conduct the following analysis. First, we present evolution of group properties over time which illustrates existing dynamics in a torrent. The evolution usually includes an initial flash crowd pattern followed by a long time in which the system experiences rather stable conditions. Second, we show the distribution of peer properties and observed group properties during life time of individual peers across participating peers in a torrent. These results illustrate that the observed performance could spread over a wide range often with no clear modes or high density regions.

Finally, we employ several statistical analysis techniques to identify key properties that determine observed performance by each peer: (i) We investigate pairwise statistical correlation between observed performance and properties and find out that the minimum outgoing bandwidth and average group content availability have the most significant correlation with the performance. (ii) Using linear regression we show that there is no linear model accurately describing the observed performance using the peer and group properties. However, in the generated linear model the median outgo-

ing bandwidth and average group content availability have the most significant effects on the model. (iii) Using data mining decision tree technique, we observed that the decision tree generated in all cases is very deep (10 fold) and large. This implies that there is no common order of importance among the peer and group properties determining the peer performance.

In a nutshell, our results indicate that: (i) there is no single property that determines observed peer performance, (ii) peer's outgoing bandwidth, available content in the group and peer arrival/departure rates appear to have the most effect on peer performance, and (iii) in a real world (actually deployed) system, there are many unexpected effects making the system unpredictable and chaotic.

This report continues in the following order. In Section 2, we present an overview of BitTorrent system explaining common terms and describing mechanisms BitTorrent uses. Section 3, categorizes the related work on BitTorrent characterization using modeling, simulation and empirical studies. Our methodology, challenges of this project, possible approaches and the specifications of the data set we have used, are described in Section 4. In Section 5, we present our characterization results. Finally in Section 6, we provide an analysis of the results.

2. BITTORRENT: AN OVERVIEW

In BitTorrent, participating peers form an unstructured and highly connected random overlay. In order to select neighbors, each peer asks the *tracker* for the addresses of a large number(e.g. 100) of participating peers. In response, the tracker sends back a list of peers known to be active. From this list, the peer will try to maintain a neighbor set of 30-50 and exchanges handshake messages and requests data within this group. Download requests may be sent to a large list but uploading is controlled by the incentive mechanism and is limited to a small set (5 peers) at any time.

At any given time, participating peers are divided into two groups; namely *leechers* and *seeds*. A seed is a peer who has already downloaded the whole file and is only helping others to download. Leechers are those still in the downloading process. They may have some segments of the file downloaded but they are still downloading other segments while contributing their up-link bandwidth to other peers who need the segments that they have. Normally a peer joins the *swarm* as a leecher with no segments available and then it gradually downloads all segments of the file and eventually becomes a seed. A seed keeps helping other peers until it leaves. In some cases, people voluntarily seed the content they have already downloaded by staying in the torrent for a longer time or rejoining the torrent at a later time.

Before joining the swarm, the user has to download a meta file called the *torrent file*. This file contains all the information needed to join the swarm and download the content. The torrent file is usually downloaded from one of the popular BitTorrent hosting and indexing web sites but it can also be distributed by other means such as e-mail. Indexing web sites, provide a list of torrents matching a particular search query which allows users to download the torrent file associated with the desired content. The torrent file is then used by the BitTorrent application to join the corresponding swarm.

A set of related files (or a single file) that are being distributed in BitTorrent as a single entity is called a *torrent*. When a torrent is initialized(seeded for the first time), the associated files are broken into smaller (*e.g.*, 256KB) parts called *segments* and a hash value is calcu-

lated for each segment. The torrent meta file is usually small (10KB-40KB) and contains technical information about the torrent including the name and size of the files in the torrent, the segment size used, the mapping of segments to files, the hash values of each segment and the tracker URL.

A tracker is a voluntary service provided by an Internet host usually under a web server in the form of a CGI program. Every peer contacts the tracker using an HTTP GET method submitting its download/upload status and possibly asks for new peers in the swarm. The tracker works as the bootstrapping node (*i.e.*, rendezvous point) by maintaining the status of all participating peers. Peers sign in with the tracker when they join and then keep sending periodic updates (usually every 30 minutes) reporting their current download and upload status. They may also contact the tracker asking for new peers as soon as they need some new peers to keep their number of peers above a minimum threshold. They also report *download completion* when they become a seed and send a *sign out* message before they leave the system.

Because trackers are invoked through a web server, the requests are usually logged in the web server's log file. This log file (*tracker log*) can be used later to analyze some aspects of the BitTorrent system (*e.g.*, [5]) because it includes progress data for each peer. Since tracker logs are the main sources of data used in this project, we will discuss them in more details in the Section 4.4.2.

2.1. BitTorrent Mechanisms

In this subsection we will describe the basic components of BitTorrent in more detail.

1. **Content delivery:Swarming** In order to provide an opportunity for every peer to contribute its up-link bandwidth to the system, BitTorrent incorporates swarming mechanism. In swarming, the content is fragmented into a large number of segments (as described before) which the participating peers exchange. After the initial segments are downloaded from the source, the peers can exchange their segments with each other. The effectiveness of the swarming mechanism also depends on the segment selection policy that will be described next.

2. **Segment Selection:Local Rarest First** To ensure diversity of the segments in different parts of the overlay and avoid situations where a peer and its neighbors all have the same set of segments, a segment selection policy is used. According to *local rarest first*, a peer examines available segments among its neighbors and tries to downloads the rarest segment first. This mechanism significantly improves the diversity of available segments and speed of *content diffusion* among the participating peers. Swarming gives peers a higher chance to contribute to the system by providing segments that other peers need.

3. **Incentive:Tit-for-tat** Almost every peer-to-peer content sharing system has an incentive mechanism to encourage altruistic peers and punish free riders. In BitTorrent this mechanism is incorporated between each pair of interacting peers and is claimed to "build robustness" for BitTorrent [2]. Each peer may download from many other peers at any time but only helps a few by providing its upload channel (*unchoking*) to those peers. By default it temporarily refuses uploading to all other peers (*choking*). To decide which peers to unchoke, each peer ranks all peers from which it is downloading, according

to the download rate it receives from each. The top four up-loaders will be unchoked as a reward of their cooperation and good resource contribution. By doing this, the peer tries to keep good uploaders satisfied and continue downloading from them. This mechanism tries to prevent free riders from obtaining a sustained service. Each peer also unchokes another neighbor selected at random. This action is called opportunistic (random) unchoking. Using this technique, peers provide an opportunity for the new comers to join the cycle of helping and being helped. Another benefit of random unchoking is that, it provides the peers with the possibility of finding new high speed neighbors. This mechanism, results in some sort of grouping among participating peers, such that high speed peers are mostly connected to each other because they usually win the competitions for providing higher rates to other high speed peers. As an effect, high speed peers are more likely to keep exchanging segments with other high speed peers.

3. RELATED WORK

Due to its growing popularity and impact on the Internet, BitTorrent has been the subject of numerous research works in the last couple of years. In this section we present an overview of the related work in BitTorrent characterization, classified by the main approach they have chosen.

3.1. Modeling and analytical studies

In this class of studies, researchers employ probability and statistics to derive mathematical models that represent different aspects of the BitTorrent protocol. Qiu and Srikant [10] provide a fluid model for BitTorrent's group size and data rates. However, Guo *et al.* [4] reject their model for not representing the initial flash-crowd in BitTorrent. They propose a more sophisticated model and verify its fitness using measurement results. Tian's model [12] uses a Markov chain connecting virtual states of a node from entering the system until download completion. The model includes arrival/departure rates, abort rate and the *efficiency* of exchanging pieces between peers depending on their level of download completion.

Modeling studies provide insight and help for better understanding of the system and relations between its parameters. However, they usually do not take into account many dynamics of the system and protocol subtleties. More importantly, validating a model requires a good understanding of "representative" behavior of the system which requires accurate measurement and is difficult to obtain. Also accuracy and applicability of the analytical models, depend on the accuracy of the assumptions they make. The modeling studies have often focused on group-level properties rather than peer-level characteristics.

3.2. Simulation studies

This group of studies develop and use discrete event simulators. They simulate events such as join, leave, pairwise connections and piece exchange without dealing with packet-level details. They often suggest mechanisms to improve the system's characteristics.

Karagiannis *et al.* [6] use simulation to show the impact of BitTorrent on ISPs. They show that the impact is significant on the ISP's uplink pipes and suggest locality-aware overlays to reduce the impact. Bharambe *et al.* [1] use simulation to characterize resource utilization and fairness in BitTorrent. They show that although BitTorrent successfully utilizes the participating peers' resources, its fairness among

peers is questionable because, in average, high bandwidth users do not contribute as much as low bandwidth ones . They suggest mechanisms such as *smart seeds* in order to improve the fairness.

Qiu and Srikant [10] and Tian *et al.* [12] also provide simulation studies. However, their works are mainly about modeling and use simulation to support proposed analytical results or to evaluate suggested improvements. Wu and Chiueh [15] perform a simulation study to evaluate the efficiency of BitTorrent as a one-to-many content distribution system compared to their centrally scheduled protocol and claim that although BitTorrent is good in minimizing the average completion time, it's far from an ideal file distribution mechanism specially in a heterogeneous environment.

Simulation studies are often closer to reality than mathematical analysis. However, they often simplify system dynamics such as churn (dynamic peer participation), variations of available bandwidth and network transient conditions that can potentially affect the results. Also, the group size in simulation studies is limited by the available hardware and software and becomes smaller as the level of details in simulation increases.

3.3. Measurement/Empirical Studies

There are a handful of empirical studies on BitTorrent. Izal *et al.* [5] use tracker logs to study group-level performance in BitTorrent. They look at the evolution of peer and seed population over time using an instrumented client to capture download and upload evolutions. They examine the overall contribution of seeds and leechers in the system and provide statistics on the number of single-session/multi-session downloads and the percentage of incomplete sessions. They also plot distribution graphs of session duration, data availability and download rate.

Karagiannis *et al.* [6] use packet traces gathered at ISP gateways to show the impact of peer-assisted content distribution systems on ISPs. Using the traces and simulation, they compare a locality-aware peer-assisted scheme with caching and current BitTorrent. They conclude that a locality-aware system will save the content providers from investing on caches and also save ISPs from buying excess uplink while users are happy downloading content at high speed.

Guo *et al.* [4] also use tracker logs together with packet traces from ISPs. They mainly use measurement results to support their proposed model for peer arrival rate. They plot measurement results together with model predictions for torrent population distribution, downloading failure ratio and download rate distribution. They also propose an inter-torrent collaboration mechanism via a tracker site overlay in order to improve content lifetime. Pouwelse *et al.* [9] conduct a measurement study on BitTorrent and SuprNova (a popular torrent search/indexing/tracker site) focusing on content availability, data integrity, flash-crowd handling and average download speed in BitTorrent. To assess the data integrity they try to insert fake files in the system with the name of popular content but the attempts are detected and blocked. They conclude that the data integrity is high, due to cautious moderators and users (of Suprnova).

Erman *et al.* [3] analyze packet traces of local BitTorrent users to characterize BitTorrent signaling traffic.

The previous measurement studies on BitTorrent presented its group-level characterization and performance analysis without examining the underlying causes. To our knowledge,

no previous study on BitTorrent has examined per-peer performance along with its root causes. Few studies have presented distribution of some aspects of per-peer performance (e.g. download time). While they provide a general notion of observed performance, true underlying causes for observed performance by individual peers are not addressed. More specifically, the effects of key factors such as overlay topology, group state and peer bandwidth have not been examined.

4. CHARACTERIZING BITTORRENT

In this section, we present our methodology in more detail. We will study possible approaches and challenges involved and report our data extraction, profiling and management methods.

Our BitTorrent characterization is performed in two different levels to capture different sets of properties: (i) In *group-level characterization*, properties of participating peers in a torrent as a whole are studied. A few examples of group properties are population, arrival/departure rate and average content availability in group. We capture these properties over time to examine their evolution pattern. (ii) In *peer-level characterization*, properties of individual peers are under investigation. These properties are captured in a time independent fashion and are used to study overall distribution of peers.

4.1. How to measure BitTorrent characteristics?

In order to characterize a deployed peer-to-peer system such as BitTorrent, there are a few measurement approaches one might employ. We will briefly discuss a few of them.

1. **Active monitoring:** In *active monitoring*, the goal is to capture snapshots of the system including different properties of participating peers with certain frequency. The measurement software should contact the tracker to get a list of participating peers. If the group size is beyond the number of peers that the tracker may provide in one transaction, several requests to the tracker may be necessary to get a list of all participating peers.

   The measurement program then attempts to contact every peer in the list and capture their properties. The time it takes to complete the snapshot is critical to the snapshot's accuracy and the measurement software should try to use its available resources to implement maximum parallelism in this process and achieve minimum time.

   This method can be used to capture many different aspects of the peers and it is the only way one may use to capture the content availability map of each peer to study *segment diversity* among the peers.

   The main shortcoming of this approach is its limited scalability. Like peer-to-peer overlay crawling mechanisms (*e.g.*, [11]), the snapshot gets more distorted as the group size gets larger because it will take longer to capture it.

2. **Running a tracker:** Hosting a BitTorrent tracker is a good way to capture the behavior and performance of all peers participating in a swarm. This approach uses the information in updates that the peers periodically send to the tracker reporting their status. The peers report their download and upload progress and the tracker will log these updates.

   The researcher may simply use the tracker logs to capture download and upload progress or implement additional features

to the system to measure extra properties concurrently. For example, it may try to measure the available bandwidth to each peer (*e.g.*, using packet train methods) and use it to calculate the utilization of of available bandwidth.

For the results to be useful, we need to examine a large group, however it can be challenging to attract many peers to our tracker. In order to attract a large number of BitTorrent users, one has to provide popular content but the popular files are most of the time copyrighted and the tracker might face legal troubles.

3. **Instrumented clients:** Running a number of instrumented clients is also a method one can use to capture the performance a peer will experience. Although information gathered by the clients can include many aspects (*e.g.*, upload and download rates, overhead traffic and topology), such information only describe a limited view and may not be representative for other peers with a variety of different conditions.(*i.e.*, access-link bandwidth, available bandwidth variability, geographical distance, different protocol implementations and configuration parameters)

4. **Tracker log:** Using BitTorrent tracker logs, one can get virtually the same information as running a tracker for a lower cost.

   If one can access the tracker logs from a popular torrent, the results can be used as a representative sample because the sample is large. The challenge here is that because most popular torrents are copyrighted material, the tracker administrators are not willing to share the log files for fears of abuse or legal prosecution of the users.

   Although the concern can be addressed by anonymizing the logs before sharing them with researchers, the administrators hardly have any incentive to spend the time to anonymize the logs and share them. Note that the time between event log records coming from the same peer is normally about 30 minutes thus it cannot compare to the fine grain sampling one can do by running a client. Nonetheless, the *global view* derived from the tracker logs is still important even with a 30 minute granularity. For these reasons, we pursued this approach in the current study. We use the BitTorrent tracker logs from three different sources as described in detail in Section 4.4.

4.2. Challenges

Measuring and characterizing a peer-to-peer system can be a challenging problem. Here we explain some of the challenges one faces in such a study.

4.2.1. Challenges in using BitTorrent tracker logs

- **Churn:** In BitTorrent, similar to many other peer-to-peer systems, peer participation is highly dynamic, *i.e.*, peers join and leave the system in an arbitrary fashion. These dynamics make it difficult to capture group level properties at a given time.

- **Coarse-grained event logs:** As explained in Section 2, the time between sending updates from a peer is usually around 30 minutes (if they need more peers, they may update more frequently). This will hide the events that occur within smaller time scales (*e.g.*, if a peer's downloading had stopped for 10 minutes). Also we cannot measure performance variability in time scales shorter than 30 minutes.

- **No topology information:** Since tracker logs do not contain any information about their neighbors, we do not have any knowledge about the overlay through which peers are connected.

  Overlay topology could potentially affect peer performance. For example, one peer may have the chance to get connected to resourceful and cooperative peers while another may not find such neighbors. Number of neighbors, as well as their resource level and available content can be important in determining a peer's observed performance. Lack of neighborhood information restricts us from exploring this possibility.

- **Missing bandwidth information:** Peers do not report their link bandwidth. In order to have a fair assessment of their performance, we need some information about their access-link bandwidth or available bandwidth. For example, a 100kbps download rate can be excellent for a 128kbps DSL link but a poor performance for a peer with a 100Mbps link.

  In some studies, researchers have used free and commercial databases, mapping IP addresses to access-link information using IP registry information and reverse DNS abbreviations. * Registry and DNS information, however, do not always translate to access-link capacity. Even for the cases where we can successfully find the access-link capacity, it may not be a useful upper bound since users can limit their maximum upload rate in BitTorrent. More than one computer might be sharing the link , other applications might be sharing the access link and multiple torrents might be downloaded concurrently. These factors may all limit the available bandwidth BitTorrent can use to download one torrent which must be our comparison reference to assess the download rate.

*c-22-22-202-114.hsd1.or.comcast.net indicates a Comcast high speed cable modem user

4.2.2. Managing a real system

In this project, we are facing some problems that are common in managing characterization data of a widely deployed distributed system.

- *Large amount of data:* Popularity of BitTorrent is a major reason that makes this characterization important. Characterizing a popular system often involves gathering, maintaining and processing large amounts of data. In this project we have processed more than 73 million lines of log files associated with more than 4000 torrents. This makes our database a large one in which conducting a simple task may take a considerable amount of time. In a large database, efficiency is a major priority and extra effort is required to optimize the database and our algorithms or queries to run faster. Nonetheless, to complete complicated tasks, a long time and reasonably fast hardware may be required.

- *Outliers:* There might be buggy or modified clients behaving in an unexpected manner. They can adversely affect measurement results by sending false reports. Data related with such buggy clients must be identified and cleaned up. For example, in our BitTorrent logs, some clients reported to download orders of magnitude larger amount of data than the file size. This problem was identified as we tried to find the file size according to the maximum download size among all peers. To solve this issue, we used the remaining counter

to check the validity of the downloaded counter and discarded a few instances for which the counters did not match.

- *Missing/incomplete data:* The measurement system might experience hardware, software or network problems interrupting measurement for some time. It is also possible that some unnoticed bug affects our results in a way that is hard to notice or fix. For instance, daylight saving time shifts can create virtual outages or out of order events in the logs.

  In our tracker logs, we faced sessions with no start record, no stop record, or out of order events. Missing an event in the logs can occur due to client misbehaving or unexpected disconnection from the network.

  In the RedHat torrent, there were about 22% of the sessions with incomplete status. For a large portion of the incomplete sessions, we have assumed the last update associated with a session to be the stop update but in some cases it is hard to determine whether the peer has left and rejoined without proper sign-out/sign-in or the logs are missing for some other reason. In these cases, we discard the incomplete session but the sessions with this problem were only about 6% of all for the RedHat torrent.

4.3. Deriving Characteristics

Now that we know about the data set and its structure, the next question is: how to derive BitTorrent properties from the tracker logs

- **Group-properties:** The following properties can be measured *at any given time* for the group of participating peers.
  - Population: Number of active peers in a torrent at a given time.
  - Churn: Peer arrival or departure rate
  - Average Content Availability: Average amount of available content for participating peers. For each peer this is the ratio of the downloaded portion over the total file size.
  - Seed percentage: The percentage of participating peers that are seeds (*i.e.*, those who have the entire file).

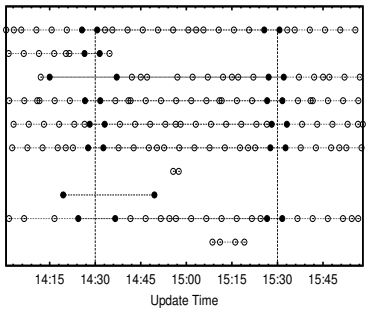
- **Peer-properties:** Besides the group parameters that are defined at any given time during the torrent's life time, we can also extract peer parameters. Peer properties capture a peer's behavior and performance during its session time. For example, average download and upload rate, contribution ratio (upload to download ratio) and geographical region are peer properties. Later we will describe our methodology for capturing peer properties according to the *session* concept.

- **Peer-view of group properties:** This class of properties are calculated by averaging each group property (*e.g.*, population, content availability, etc.) during a peer's life time. Peer-view of group properties can also be used as a potential factor affecting peer performance.

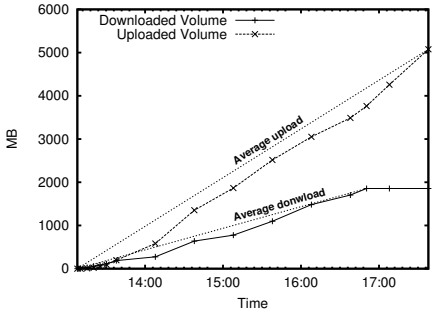
4.3.1. Measuring group properties

We pursued two different approaches in measuring group properties. Here, we present both methods along with their cons and pros.

Window-based approach: To capture group properties, we define a measurement window of length τ and measure each parameter within this time window. For example, for average download and upload rates, we scan through all active sessions in the measurement



(a) Sampling method for group properties



(b) Calculating average download and upload rates

Figure 1. Extracting peer-level and group-level characteristics

window and by comparing the first and last updates coming from each peer, we calculate average incoming and outgoing bandwidth for that peer. Then we record the average of all these averages as the download and upload rate associated to that measurement window.

For some other parameters that can be defined using only one update like content availability, we only consider the last update from each peer within the window and calculate the average content availability as the mean of all values captured as described above. This method gives an average of the group status in each time window and can be used to study the evolution of different group level parameters over time.

In order to check the accuracy of this method, we compare total average download and upload rates and the results have shown that average download rate was sometimes more than 50% higher than the average upload rate. We expected the two to be equal because the system is closed and every downloaded byte for one peer is an uploaded byte for another. The difference showed an error in the approach which was not too hard to see. We averaged all sessions with equal weight but that cannot capture a clear picture of the system. A one minute long session is not as important as a one hour long session. To correct this distortion, we weighted the averaging by an estimate of the session length within the time window which is the time between the first and the last update coming from that particular peer within the time window. Applying this weighting scheme, improved the results and there was only 5% difference between download and upload rate. To explain the cause of error in the last approach we should note that there are many short-lived sessions that join the system and leave within a few minutes. These sessions usually download much more than they can upload because they don't have any content to upload. These short-lived peers bias the average download rate to be higher than upload rate. Also the average content availability was higher when we used weighted averaging rather than simple averaging which can be explained with the same concept as above.

Selecting the right window size is a key issue in this method. Obviously, the window must be long enough to include at least two updates from each peer (to capture progress). But also considering the weighted averaging scheme, the

weight of each session's parameters in averaging, is the time between the first and the last update within the measurement window. Ideally, we want this to be equal to the time the peer has been active in the time window. However, considering the updates granularity, we notice that the mentioned weighting coefficients can be shorter by up to 30 minutes. For example, the session might be active during the whole measurement window while the update has come after 30 minutes. This will introduce an error in the weighted averaging approach.

The relative error can be reduced by choosing larger measurement windows in which case the absolute value of error remains the same (up to 30 minutes) but now it's compared against larger values. However, the measurement window size determines the resolution of the measurement and certainly we want to keep it as small as possible. We chose the window size of 4 hours which is 8 times the maximum value of the window error and reduced our error indicator to less than 5%.

Sampling approach: To avoid the shortcomings of window-based approach, we develop ways to cancel the bias in averaging which yielded to the sampling point method. In this method, we sample the group at evenly spaced intervals to capture the status of the group at that particular time as shown in Figure 1(a). In this figure, each circle represents an update and each horizontal line connects updates from the same peer representing a session. The two vertical lines represent the sampling times and the filled circles are the updates that have been used for that particular sampling. The method is slightly different for different parameters. For some parameters like group population, we simply count all sessions started before and finished after the sampling time. For parameters like download and upload rate, we use the last update from every peer happening before the sampling point and the first update after the sampling point to estimate the rate at the sampling point. This is depicted in Figure 1(a). For some other parameters we also use interpolation. For instance, for content availability, we interpolate between the values of the two updates surrounding the sampling point assuming a linear progress in time. The sampling points can be chosen arbitrarily close to each other but if the distance is shorter than 30 minutes (normal update interval) then it can cause oversampling which results in extracting and processing more data than the information they carry.

The sampling approach generally gives a better resolution of the evolution but the downside is that it misses the variations happening between the sampling points.

4.3.2. Deriving peer properties

Session approach: We define a session, as the set of updates coming from a certain peer in one appearance in the system. Normally a session starts with a *sign in* event in the tracker log and continues with periodic updates, it may also include a *download completion* and finally ends with a sign out event. However, due to misbehaved or disconnected peers any of these events might be missing. Each peer has a unique peer-id as well as an IP address and port number that can be used for identification altogether. In the following we introduce the peer properties and briefly describe how we capture each. In Figure 1(b), cumulative upload and download counters for a particular peer are plotted against time. The slope of each segment of the download graph shows average download rate during that interval.

- *Average download/upload rate:* To calculate each peer's average bandwidth, we use the first and last update from that peer and divide the increase in uploaded and downloaded bytes to the session time as

shown in Figure 1(b) For the incoming bandwidth or download rate, it should be noted that downloading only happens up to the completion point, therefore the time from session start until the download completion is used instead of the total session time. Since the focus of this study is the peer performance which can only be defined during the downloading time, we do not take into account the upload status after download completion either. Therefore both upload and download data are associated to the downloading time (before completion).

- *Bandwidth statistical specifications:* Besides average value, we also care about range and variations of download and upload rates. We record the maximum incoming and outgoing bandwidth (download and upload rate) together with the length of the interval in which the maximum has happened. To capture the statistical specifications of the bandwidth range and variations, in addition to the maximum, we record standard deviation as well as 10th, 50th and 90th percentile.

- *Contribution:* In each time interval between two updates, we also capture a *contribution* value which is defined as the amount of uploaded bytes divided by the amount of downloaded bytes by the same peer during the same interval. Similar to upload and download rates, for *contribution* also we capture and record statistical specifications including maximum, the percentiles and standard deviation.

- *Region:* We used GeoIP [7] database to map every peer's IP address to its geographical location. To simplify the process, we only recorded a number in the range of 1-6 to represent the region in which the peer is located in the world. Numbers 1 to 6 represent: North America, Europe, Asia, South America, Oceania and Africa respectively.

A reference point is required for each peer to make a reasonable comparison between the download performance of different peers. We use maximum observed incoming/outgoing bandwidth as an estimator of the peer's available incoming/outgoing bandwidth and define incoming/outgoing *bandwidth utilization* as the average bandwidth divided by maximum bandwidth in each direction. The maximum value is the best estimate we have for the available bandwidth but it is actually only a lower bound to the access-link bandwidth. This value is inaccurate for the following reasons: (i) As mentioned before, the tracker logs are rather coarse grained. Normal interval between two updates from the same peer is 30 minutes. Maximum download rate is an average over such an interval and obviously can be a result of many variations. Intuitively we expect it to be considerably lower than the real maximum rate specially when the interval in which it has happened is long. (ii) It should also be noted that the available bandwidth can be variable during the download time. Many events from the user side or the network side can effect the available bandwidth. Multiple users can start or stop sharing the link, multiple applications can start or stop sharing the link (*e.g.*, downloading a file) and multiple torrents can be downloaded at the same time.

We could also estimate a peer's access-link bandwidth using IP registry and reverse DNS information as described before. However, we believe this approach does not give a more reliable estimate of the available bandwidth either. When access-link bandwidth has been correctly registered, it can only project an upper bound for the available bandwidth. Sharing possibil-

Community	#Torrents	Start Time	End Time	#Sessions
Red Hat	1	3/2003	8/2003	170814
Debian	1599	2/2005	3/2005	1268003
3D Games	2585	8/2003	12/2004	4416738

Table 1. Studied data-set information

ities, user configured rate limitations, network bottlenecks and congestion may all cause available bandwidth for a BitTorrent client to be less than the access-link bandwidth.

Peer-view of group properties: In order to study the effect of group properties on the peer's observed performance, we need to present it from peer's point of view. Earlier we described that we capture group properties (*e.g.*, population, seed percentage, ...) on sampling times. We define peer-view of each group property as the average value of that property during a peer's downloading time. We simply average over all samples capture during peer A's download time and call the result, peer A's view of the group properties. These properties include *population, seed percentage, average content availability, arrival and departure rates.*

4.4. Data Set

In this section, we present a description of our data set used in this study. The data includes three sets of tracker logs from three different sources:

1. Red Hat: Tracker log files from one torrent namely the CD images of Red Hat Linux 9.0 distribution.
2. Debian: Tracker log files for 1599 different torrents related to Debian Linux distribution.
3. 3D Games: Tracker log files for 2585 different torrents from 3D games community.

Table 1 summarizes the specifications of the three tracker log files used in this study.

4.4.1. Cleaning/Sanity checking

Data gathered from real measurement may contain inconsistencies that must be fixed before processing.

It is likely that the data includes inconsistent or incorrect parts. To derive BitTorrent characteristics accurately, we need to detect and address each problem or inconsistency in data, accordingly.

- As explained earlier, our study is based on sessions. A large portion of the sessions are not completely detected by our detection method because of missing updates. For some sessions, we cannot find a sign-out record and they just vanish from the system for some unknown reason. This can be due to software crash, link disconnection (very probable for dial-up), or a misbehaving client. We can not discard these *incomplete* sessions since they constitute a significant portion. We use the last message sent from these peers as their sign-out message and calculate their properties accordingly. For some other sessions, we have several sign in messages often in a short time without signing out. This can also be due to a bug in the client software or the unlikely event of repeated disconnection from the network and signing in again in a short time.

217.160.111.64 - - [31/Mar/2003:12:52:48] "GET /announce?**info_hash**=E%e9%c6%1d%dc%7eA%cd%eb%97%c8%85%dc%26M4%db%11%18%1d **&peer_id**=%d6%be%9f%88%28C5%eb%1c%cdI%98j%c5H%80%5d%3bJ%99**&port**=52000 **&ip**=134.106.11.159**&uploaded**=0**&downloaded**=0 **&left**=1855094951**&event**=started HTTP/1.0" 200 162

Table 2. Sample event log from Red Hat torrent tracker

time	Event time and date
IP & port	Real peer IP address and port number
peer_id:	A unique 20 Byte ASCII string chosen randomly by each peer
info_hash:	A 20 Byte SHA1 hash of torrent info uniquely identifying a torrent
uploaded:	total number of bytes uploaded by the peer in the current session
downloaded:	total number of bytes he peer has downloaded in the current session
left:	total number of bytes remaining until the peer has completes download
event:	is one of the strings: 'started', 'completed', 'stopped' or non-present

Table 3. Event log fields description

- There are many short sessions in which a peer does not have enough time to participate in the system and help others. A peer only sends an update every 30 minutes which means that we don't have enough updates from a short session to capture its progress. Moreover, during the session time, it experiences a poor performance. Such sessions do not represent our study target because they leave the system before the BitTorrent mechanisms start to work by participating in the swarm. Therefore, we do not include sessions shorter than 40 minutes in our performance characterization. 40 minute should provide enough time for the peer to participate in the swarm.

- Because we use updates coming from a peer to capture statistical specifications of download and upload rate, we need a minimum number of updates for each session for the statistical parameters (like percentiles) to be meaningful. We believe that if the number of updates from a peer is less than 5 updates, then they are probably too few for any meaningful statistical analysis. Therefore we discard sessions with less than 5 updates during their downloading time.

- In order to check consistency and integrity of the data gathered, in our windows based approach, we compared the overall average upload rate and download rate and found them to be within 5% of each other using the weighted averaging method described in Section 4.3.1.

- For each session, we checked to make sure we had monotonically increasing download and upload byte counts. This was always true except a case in which daylight saving time shift occurred.

4.4.2. Tracker log description

A BitTorrent tracker is usually a CGI program running through a web server. The communication between the BitTorrent peers and the tracker is an extension of HTTP . Peer requests and updates use GET method passing all their data to the tracker program using the usual format of parameter passing in HTTP GET.

A sample line from one of the log files is presented in Table 2 and the fields are described in Table 3.

4.4.3. Peer-level Table

In order to use statistical methods to characterize peer performance, we made a table for each torrent, in which every row represents a session in that torrent and every column is either a *peer property* or a *peer's view of a group property*. We tried to make this table comprehensive and inclusive of all parameters that *could possibly* affect a peer's performance. To capture group level parameters we used the sampling method described earlier to create a table showing evolution of all group properties in time with 40 minute resolution. To get the peer view, we grabbed all rows of the evolution table falling in the peer's lifetime and averaged over each column. For instance, by averaging over all group population values captured during the life time of a peer we calculate average group population from that peer's view and record it in the respective row of the peer-level table. The table includes the following fields:

- The peer's identification information including IP address and port, peer_id, etc.
- Statistical parameters of download rate including average, maximum, 10th, 50th and 90th percentiles and standard deviation.
- Statistical parameters of upload rate (same as above).
- Statistical parameters of bandwidth contribution (same as above)
- Peer-view parameters of group properties including population, seed percentage, content availability, arrival/departure rate.

5. GROUP- & PEER-LEVEL CHARACTERISTICS

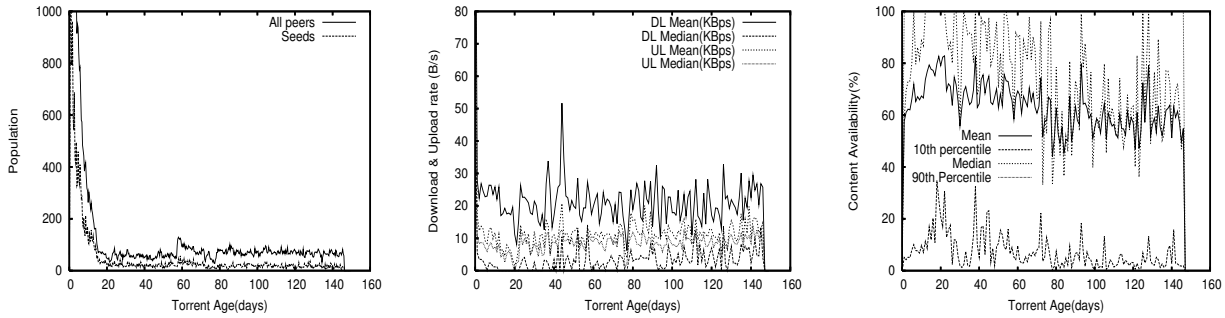
In this section, we provide the results of BitTorrent characterization in three parts. First, we present the evolution of group-level properties over time for different torrents. Second, we report the distribution of different peer-level properties across all sessions participating in a torrent. Finally, the distribution of peer-views of group level properties are discussed.

5.1. Group-level characteristics:

We are interested to see the evolution of all participating peers as a whole over time. It is insightful to know how the properties of the peers as a group changes over time and how these properties might affect each other. For example, it is important to know if there are times when there is little content available in all peers in average or if the average download rate depends on content availability or seed percentage. The group properties evolution study gives us a good understanding of what happens in a BitTorrent swarm from the beginning till the end.

We present these properties in two different time scales: (i) The *sampled view* shows the whole time available in the tracker log we use. We have reduced the data presented to two data points per day to make the graphs more readable. (ii) The *zoom-in view* provides a detailed view of the first 10 days of the tracker log including all the samples available (one every 40 minutes). For most torrents, this interval is also the first 10 days of the torrent life and depicts the initial large demand.

For most popular torrents, we observe a flash crowd pattern at the first hours or days and then the stabilization phase begins which can last for months depending on the content and finally the finishing part is observed. In sampled-view graphs, (*e.g.*, Figures 2(a), 2(b) and 2(c)), presented data points are 12 hours apart to make the graphs more readable despite noisy behavior of the phenomena. Whereas in zoom-in graphs (*e.g.*, Figures 3(a), 3(b) and

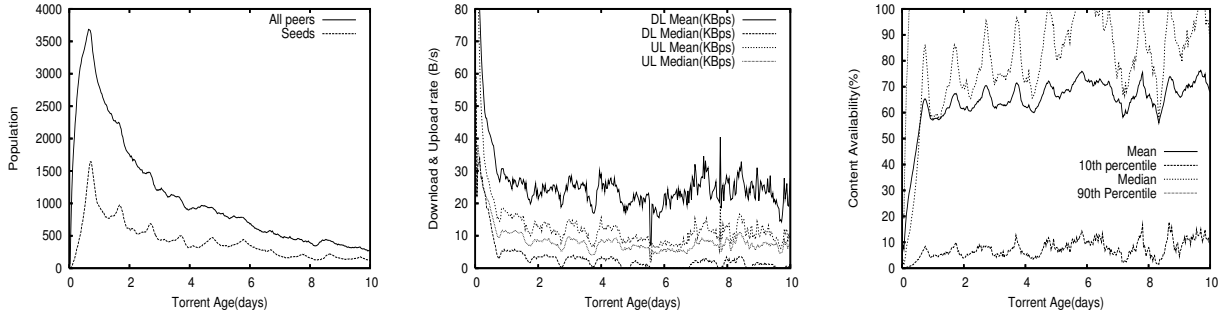


(a) Population

(b) Download and upload rates

(c) Content availability

Figure 2. Evolution of group properties for the RedHat torrent during 150 days using 2 samples per day (sampled view)



(a) Population in the first 10 days

(b) Rates in the first 10 days

(c) Content availability in the first 10 days

Figure 3. Evolution of group properties for the RedHat torrent during the first 10 days using one sample every 40 minutes (zoom-in view)

3(c)), we show a 10-day interval with all data points available (40-minute resolution).

- *Population and number of seeds:* Figures 2(a) and 3(a), show the sampled and detailed view of the total population and number of seeds for the RedHat torrent. During the first two days the torrent is announced, a flash crowd increases the population of participating peers to 3500. After the initial peak, the population rapidly decreases within 20 days to less than 100. Past this phase, the population becomes rather stable with only minor variations due to time of day and day of week effect on user demand. After this interval, the population quickly reduces to zero.

  These figures also show the number of seeds. The most important point is that the number of seeds follows a pattern similar to the total population with a small lag. This indicates that peers successfully turn into seeds and the large volume of new comers arriving in a flash crowd pattern, does not disrupt BitTorrent's process

of content distribution.

- *Download and upload rates:* Figures 2(b) and 3(b) depict mean and median download and upload rates for the RedHat torrent in sampled view and zoom-in view respectively. These figures show that not only the download and upload rates are not degraded at the time of flash crowd, they are actually higher at the beginning and then drop back to their stable status after 1 day. The higher download rates in the first day can be due to the capacity of the first day downloaders. Intuitively, note that most people waiting for a new release of RedHat and downloading it at the first day it is available, are likely to have high speed connections at academic and research institutions.

  We recall that download rate is only observed among leechers whereas upload rate is captured across all peers. This explains the difference between mean download and upload rate. The large difference between mean and median download rates is due to the tail of the download rates distribution, *i.e.*, although most of peers are low bandwidth (50% are below median), a small number of peers are very high bandwidth. We further elaborate this issue in subsection 5.2 presenting the distribution of download and upload rates.

- *Content availability:* Average amount of available content among participating peers in a torrent represents average ability of peers to serve content and thus contribute their bandwidth. In one hand, as peers gradually progress in downloading the content, the average content availability increases. However, when peers with high content (especially seeds) depart or new peers join the system, the average content availability decreases. Figure 2(c) shows that after the initial flash crowd (*i.e.*, first day), the content availability reaches a stable state with daily and weekly fluctuations. We also notice that the median content availability is always higher than the mean.

  To explain this, we should observe the content availability distribution. One might expect a uniform distribution in steady state, however since many peers leave the system just a few minutes after they join, it is more probable that a randomly chosen peer has very low content availability. Examining figures 3(c) and 3(b) illustrates that low average content availability in the initial flash crowd, does not result in lower download rates. This indicates that even peers with small amount of content can contribute their outgoing bandwidth This is mainly due to the diversity of available segments among peers.

5.2. Peer-level characteristics

Group-level characterization provides an insight on the evolution of the group properties over time. Here we present a time-independent view of the participating peers in the three torrents described earlier. Peer properties are characteristics of individual peers such as average download rate, maximum upload rate and average contribution ratio. Using distribution graphs for peer properties, we can observe any potential modes in the peer property distribution that we might use to divide the data set into different sub-groups. Such grouping can help in separating peers with different characteristics and study each group separately.

We present the CDFs for the three torrents in one graph to provide ease of comparison.

Average download rate: Figure 4(a) shows the distribution of average download rate (or incoming bandwidth) among leechers in the three

torrents. The graph shows that: (i) There is no obvious mode or critical value in the distribution to be used for grouping, (ii) The peers in RedHat torrent observe a higher download rate comparing to the other two torrents. To explain this, we should consider the difference in the size of the three torrents. The RedHat torrent is approximately $1.8GB$, almost three times larger than the Debian torrent and six times larger than the Gaming torrent. Since users with low speed links are less likely to download very large files (simply because it takes too much time), we expect to see more high speed users in torrents with very large file size.

Maximum download rate: Figure 4(b) presents the distribution of maximum download rate among leechers. The graph shows that: (i) Similar to what we observed for average download rate, peers in the RedHat torrent observe a higher maximum download rate and (ii) Specially for RedHat torrent, in a few points there are slightly outstanding regions showing a higher density of peers in that regions.

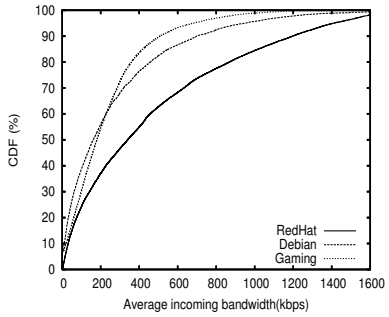
We should notice that the maximum download rate, if measured in a short time interval and the total download time has been long enough can be a good estimator of the link bandwidth. Obviously, this can only be true if no other factor is limiting the link bandwidth. Therefore, observation (i) can also support our observation in the previous graph showing higher average download rate for the RedHat torrent meaning that the access-link bandwidth of the peers in RedHat torrent have been higher comparing to other torrents.

We also notice that the slightly outstanding points in Figure 4(b) correspond to common access link technologies and commonly available commercial Internet service provisions. The outstanding points are approximately at: 256kbps, 512 kbps, 768kbps and 1700kbps. If we have a good estimator for the access-link bandwidth, we would expect to see almost all the values distributed among these values however we can only observe very small increased density in these regions in maximum download rate graph and this indicates the error in estimating link bandwidth using maximum download rate.

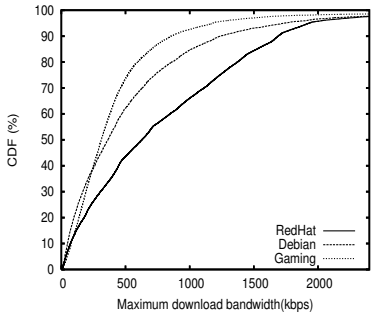
Average upload bandwidth: Figure 4(c) shows the average upload rate for all participating peers. Comparing this graph with 4(a) indicates that generally download rates are much higher than upload rates. For instance, 50% of peers in RedHat torrent observe a download rate of 400 kbps or more while only 3% of peers achieve such upload rate.

Session length: Figure 4(d) depicts the distribution of session length among all participating peers for the three torrents in logarithmic scale. The session length is the time interval between a peer's arrival until its departure and includes downloading time and *lingering time*(*i.e.*, seeding time). This graph shows that the RedHat and Gaming torrents have roughly similar distribution patterns which follow a linear distribution from about 10 seconds up to 100000 seconds (more than 24 hours). For the Debian torrent there is a high density region between 200 and 1000 seconds.

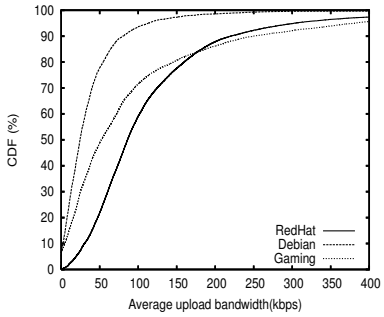
Lingering time: Lingering time is the time interval since the peer completely downloads the file (all segments) until its departure. During this time the peer is only contributing its uplink bandwidth to help other peers. Lingering might be a voluntary action of the user or the time before the user notices download completion and stops seeding.



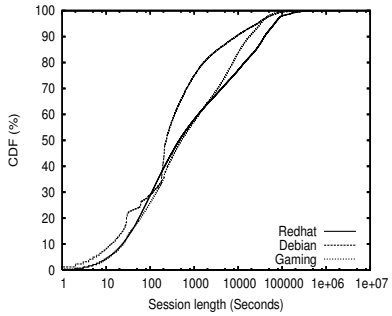
(a) Average download rate



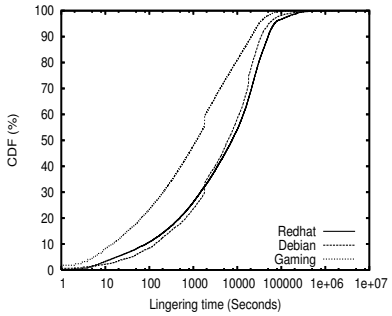
(b) Maximum download rate



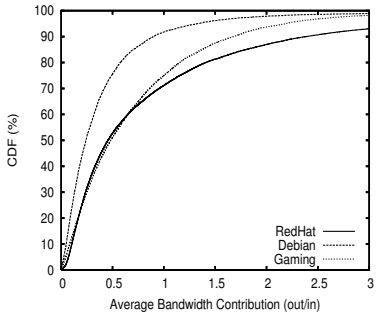
(c) Average upload rate



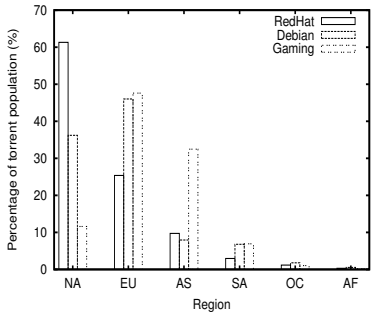
(d) Session length distribution



(e) Peer lingering time distribution



(f) Average bandwidth contribution



(g) Regional distribution: NA: North America, EU: Europe, AS: Asia, SA:South America, OC: Oceania, AF: Africa

Figure 4. Peer-level properties for three torrents

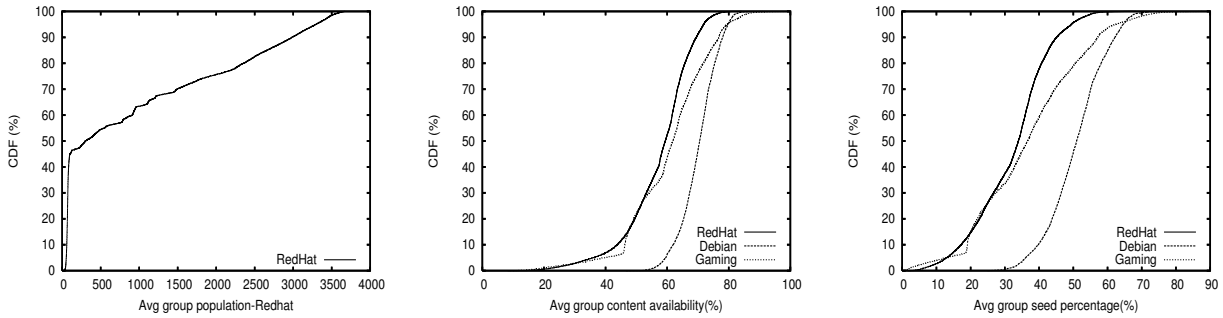
Figure 4(e), shows the distribution of lingering time for sessions who have eventually completed downloading. The graph shows that about 50% of completed downloaders stay in the system for more than 2 hours seeding the content for the RedHat and Debian torrent while for the Gamin torrent, the median lingering time is less than 20 minutes. There is a small number (about 3%) of peers with lingering time of more than 100'000 seconds (27.7 hours) for the RedHat and Debian torrent. This graph illustrates that a good level of altruism among participating peers exists which can explain why the average content availability is almost always above 50%.

Average contribution: Average contribution is the average amount of upload to download ratio during a peer's download time. Figure 4(f) provides the distribution of average contribution among all leechers for the three torrents. The graph shows that the calculated contribution is different for the three torrents. For Debian torrent, approximately 90% of peers have a contribution ratio of less than one and only 10% of peers upload more than they download. This is only possible when the altruistic 10% contribute very much to make up for the 90% who download more than they upload. For the other two torrents also most of the peers have a contribution ratio of less than one. The distribution is heavy tail for all three torrents and shows that the altruistic peers have a key role in the system performance. This heavy tail distribution can also be interpreted as unfairness in BitTorrent, however, since the high contributors which form the tail of the graph can be voluntary seeds who stay in the system for a long time after they have downloaded the whole data, we don't have enough evidence to make such a claim.

Regional distribution: Figure 4(g) shows the geographical distribution of all participating peers in the three torrents. We can observe that the RedHat torrent is very popular in North America with more than 60% of peers from this region. On the other hand, the Debian torrent is more popular than RedHat in Europe and Asia. For the Gaming torrent we can observe that number of downloaders from North America is much less than Europe and Asia. It may suggest that legal issues limit the number of users who download copyrighted material in North America.

Peer-view of group properties: In Figure 5, we observe the distribution of group status parameters from peer view. As explained before, peer-view parameters are calculated by averaging a group property over the life time of a peer. They are used in the statistical analysis to reflect the effect of group status on peer performance. Figure 5(a) shows the peer view of group population in the RedHat torrent. This graph shows two different groups of peers; about 45% observing a small population of less than 100 and 55% observing higher populations almost uniformly distributed up to 3600. To better understand this graph we should also consider the population evolution graph in Figure 2(a). The comparison suggests that the two groups are separated by their arrival time; the top 55% have arrived during the initial 20 days of high popularity of the torrent while the remaining 45% have arrived during the rest of the torrent life. This point implies the importance of the initial phase because the majority of peers have joined during that time despite its shortness (20 days in 5 months).

Figure 5(b) depicts the distribution of average group content availability during a peer's session time across all participating peers for the three torrents. We can observe that most peers experience a high level of available content for example in RedHat and Gaming torrents, 75% of peers observe an average content availability of at least 50% and about 99% of peers average content availability is at least



(a) Group Population

(b) Group content availability

(c) Group seed percentage

Figure 5. Peer-view of group properties distribution

20%. This indicates that lack of content almost never happened in the torrents we examined. This graph also shows that the average available content has been higher in Debian torrent such that almost all peers observe an average content availability of at least 50%.

Figure 5(c) displays the distribution of average seed percentage in the torrent across participating peers. Seeds only provide content and contribute bandwidth to other peers but they do not download anything. Therefore the percentage of seeds is an important metric of the system's available resources including content and bandwidth. This graph shows that the seed percentage is higher in Debian torrent comparing to the other torrents. Higher number of seeds can also explain higher content availability for this torrent that we observed in Figure 5(b). Among peers in Debian torrent, almost all of them observe an average of at least 30% of peers to be seeds which is a high level. Even for the RedHat torrent though, about 98% observe a average seed percentage of at least 10%. Remember that a leecher may download all the segments it needs from other leechers and it may mot need to contact a seed at all. Therefore even a seed percentage of 10% should be enough for the torrent to work smoothly.

6. STATISTICAL ANALYSIS

In this section, we try to answer the main questions motivating this research work. As a reminder, we wanted to answer the following questions: (i) *What are the main factors affecting observed performance by individual peers?* and (ii) *How can we quantify the effect of these factors on the observed performance?*

The results in section 5 deepens our understanding of the group-level and peer-level characteristics of BitTorrent. However, they did not reveal the key factors that affect the peer performance. Now we use different statistical methods and tools to detect and evaluate any relationship between those properties and the observed performance by the peers.

6.1. Performance Metrics

Defining a metric to capture the observed performance of a peer is a challenging task. Average download rate may seem an obvious performance metric. However, a correct evaluation of a peer's performance is only possible by comparing its download rate to its *incoming*

bandwidth[†]. For example, assume the incoming bandwidth of peers A and B are 128kbps and 10Mbps, respectively. In this scenario, a download rate of 100kbps is considered a good performance for peer A but a poor performance for peer B.

An ideal performance metric would be average bandwidth utilization defined as

$\frac{Avg.\ Download\ Rate}{Incoming\ Bandwidth}$.

As we discussed in Section 4.2.1, we cannot accurately capture a peer's incoming bandwidth. Maximum download rate can be used as a lower bound estimator for the available bandwidth with certain conditions and reservations. In the following analysis, we use the following as a performance metric.

$Inbw\ Utilization = \frac{Avg\ Download\ Rate}{Max\ Download\ Rate}$

Stability of the download rate can also be used as a performance metric. Note that the download rate variation is *inversely* related to the utilization of incoming bandwidth. Observing less variation in download rate means that the average download rate is closer to the maximum download rate. *Standard deviation of download rate* represents the level of variations in download rate but it also depends on the scale of the download rate itself. For example, for a high speed peer, because the download rate is generally higher than that of a low speed peer, the amplitude of the variations are also likely to be larger. In order to perform a fair comparison, we need to normalize the standard deviation. Therefore, we use normalized standard deviation of download rate as an inverse performance metric. Normalized standard deviation is calculated as:

$\frac{Download\ Rate\ St.\ Dev.}{Avg.\ Download\ Rate}$.

[†]Incoming bandwidth is the amount of downlink bandwidth available to BitTorrent and may be limited by the link bandwidth, other applications or user configured restrictions.

To perform the statistical analysis, we use S-Plus statistical package Version 7.0.0 for Linux which provides tools for data manipulation and statistical processing.

6.2. Sessions Table

Before we describe different statistical analyses we have performed, we should present the basis of these analyses. We build a table of all parameters that may possibly affect the peer performance. These parameters include peer-level properties and peer-view of group level properties as described in Section 5.2. Each row of the table is associated with a session and each column represent a peer property or peer-view of a group property. We also add two columns for the two *performance metrics* we defined earlier. We call the resulting table the *sessions table*. A sample part of this table is presented in Table 6.2.

We can consider the two columns on the left showing performance metrics as the output and the rest of the columns as the input to the system. Our goal here is to determine relationships between system input and output parameters. In the next subsections, we describe the statistical methods along with the results of each method.

6.3. Scatter-plots

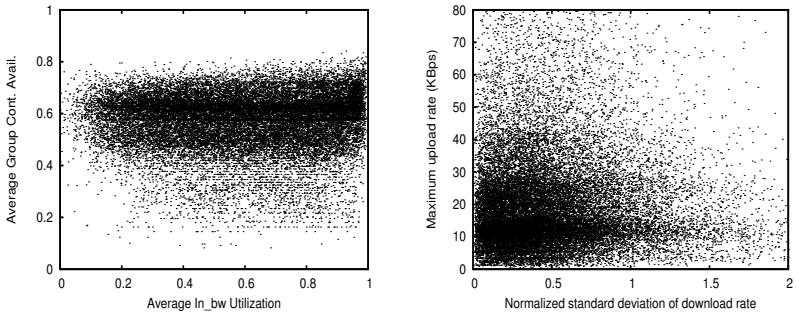
As the first effort, we use scatter-plots to visually find any relationships between the performance metrics and the properties.

In a scatter-plot, two statistical parameters are associated to X and Y axes and each measured date corresponds to a dot in the graph. The density and the pattern of the dots may discover certain relationships between the two parameters.

In Figure 6(a), the X axis represents Average incoming bandwidth utilization and the Y

Download rate						Upload rate			Cont.	Peer view of group					Perf. Metrics	
Avg	Max	10p	50p	90p	Sd.	Avg	Max	...	...	pop	Cnt av	Seed prc	Arr	Dep	Util.	Norm. Sd.
29k	32k	0	30k	32k	15k	19k	22k	...	...	522	0.35	0.40	20	15	0.90	0.51
256k	463k	0	202k	463k	159k	264k	456k	...	...	643	0.15	0.1	100	10	0.55	0.62
...																

Table 4. Sample lines from the sessions table. Properties associated with two sessions are provided.



(a) Avg. in. bw util. vs. Avg. cont. avail.

(b) Normalized st. dev. of download rate vs. Maximum upload rate

Figure 6. Sample scatter-plots of performance metrics vs. different properties

axis shows peer-view of average group content availability. The pattern of dots shows no obvious correlation between the two parameters in this figure. As another example, in Figure 6(b), the X axis is normalized standard deviation of download rate and the Y axis is maximum upload rate. In this figure also, we cannot detect any obvious correlation between the two parameters.

Next, we use more rigorous statistical tools to discover correlations between performance metrics and the properties.

6.4. Correlation

In this analysis, we use Spearman's rank correlation test [‡] to examine correlations among different pairs of columns from the sessions table. Remember that this table includes all peer properties, peer-view of group properties and the performance metrics for all sessions.

Table 6.4 presents the results of the correlation analysis for the RedHat torrent.

According to this table (the two columns on the right), the two performance metrics show opposite correlation with the listed parameters. (*e.g.*, Separate calculation showed a Spearman's rank correlation coefficient of -0.81 between the two performance metrics in the RedHat torrent)

[‡]Spearman's rank correlation coefficient, named after Charles Spearman and often denoted by the Greek letter ρ (rho), is a non-parametric measure of correlation that is, it assesses how well an arbitrary monotonic function could describe the relationship between two variables, without making any assumptions about the frequency distribution of the variables. [14]

Parameters	avg.inbw	max.inbw	inbw.10p	inbw.50p	inbw.90p	inbw.sdev.norm	inbw.util
avg.outbw	0.55	0.51	0.52	0.53	0.53	-0.18	0.21
max.outbw	0.48	0.52	0.39	0.44	0.51	0.02	0.01
outbw.10p	0.51	0.42	0.58	0.51	0.46	-0.32	0.32
outbw.50p	0.52	0.49	0.51	0.52	0.51	-0.18	0.21
outbw.90p	0.53	0.55	0.46	0.51	0.55	-0.06	0.10
outbw.sdev	0.33	0.45	0.17	0.29	0.41	0.26	-0.21
avg.grp.pop	0.10	0.18	-0.00	0.07	0.16	0.20	-0.13
avg.grp.seed.perc	0.03	0.03	0.00	0.03	0.03	0.00	0.02
avg.grp.cont.avail	0.03	0.02	0.02	0.04	0.02	-0.03	0.05
avg.grp.dep.rate	0.09	0.17	-0.01	0.06	0.15	0.19	-0.12
avg.grp.seed.dep.rate	0.07	0.15	-0.02	0.04	0.13	0.20	-0.13
avg.grp.arr.rate	0.10	0.18	0.00	0.07	0.16	0.19	-0.12

Table 5. Spearman's Rank Correlation coefficient results for RedHat torrent

An expectable result which is observed in this table is the strong correlation between the download and upload rates.

This phenomenon can be explained in two ways: (i) The BitTorrent incentive mechanism (tit-for-tat) tries to provide higher download rates for those peers providing higher upload rates. This high correlation between average download rate and average upload rate shows how successfully the incentive mechanism is performing. (ii) Given typical available access links, links with higher incoming bandwidth often have higher outgoing bandwidth. This effect is independent of the application and does not imply anything about BitTorrent behavior.

Generally, the table shows that the more a peer contributes, the better performance it observes. Specifically, 10th percentile of outgoing bandwidth has the highest correlation with both performance metrics. Comparing the correlation of 10th, 50th and 90th percentile with the performance metrics, shows that the 10th percentile is more important than the other two. Meaning that, the minimum contribution a peer has provided to other peers is more important in its observed performance than the median contribution. In other words, a peer should *always* provide a good contribution to receive a good performance.

The table also shows that group status parameters are also effective in the peer performance but with smaller effect than the peer's outgoing bandwidth. We observe that group population negatively affects the peer performance as well as churn parameters (arrival and departure rates). The negative effect of churn (dynamic peer participation) on peer performance can be explained as follows. Downloading process is interrupted every time an active neighbor leaves the system and the peer has to find a new peer to download from. The arrival and departure rates therefore can negatively affect performance by interrupting the download.

It is also important to notice that the parameters with insignificant correlation to performance metrics. The table shows that maximum upload rate does not affect a peer's performance. Also average group content availability and average seed percentage do not appear to have a significant impact on the performance a peer observes. Considering the distributions of these two parameters in Section 5 we can see that the average content availability and seed performance are always in good status. Therefore the absence of any statistical correlation here does not show they are not important but, they are always above the minimum required for peers to find enough content to download.

6.5. Linear Regression

Linear regression is a classic statistical tool to establish a linear model using a set of input

Model	R-square	outbw.50p	log(outbw.50p)	contrib.50p	avg.grp.pop
1	0.176	0.0022,0,0.075	–	-0.0729,0,-0.22	-0.0440,0,-0.132
2	0.189	–	0.0482,0,0.301	-0.0705,0,-0.21	-0.0510,0,-0.187
3	0.178	0.0022,0,0.075	–	-0.0726,0,-0.22	–
4	0.191	–	0.0462,0,0.289	-0.0705,0,-0.21	–
5	0.190	–	0.0460,0,0.283	-0.0705,0,-0.21	–
-		log(avg.grp.pop)	avg.grp.seed.perc	avg.grp.cont.avail	avg.grp.arr.rate
1		–	-0.0667,0.18,-0.0453	0.1042,0.07,0.0875	0.0002,0,0.110
2		–	0.0210,0.67,0.0142	0.0485,0.40,0.0407	0.0002,0,0.110
3		-0.0366,0,-0.047	-0.0358,0.46,-0.0243	0.2019,0.00,0.169	0.0003,0,0.165
4		-0.0396,0,-0.051	0.0673,0.17,0.0457	0.1196,0.4,0.1	0.0003,0,0.165
5		-0.0400,0,-0.052	–	0.1955,0.00,0.164	0.0003,0,0.165

Table 6. Linear regression results for the RedHat torrent

parameters to predict the value of another parameter using empirical data.

In this subsection, we use linear regression to build a linear model for each of the performance metrics as a function of the peer and group properties listed in the sessions table (refer to Table 6.2). The coefficients provided by the linear model can be used to determine the significance of effect of each parameter on the performance metric. By comparing the coefficients, one could compare the importance of the corresponding parameters.

It is important to note that different parameters have different scales which in turn affects their corresponding coefficient provided in the model. For example, if we have average upload rate as an input parameter and we use *bps* as the unit, the values of this parameter will all be larger than 1000 and therefore the corresponding coefficient will be small, whereas using *kbps* all values will be 3 orders of magnitude smaller and therefore the corresponding coefficients will grow 3 orders of magnitude larger to keep the model intact. Therefore comparing the coefficient in the model resulting from linear regression is not a correct way to assess significance of each parameter, but also we should take into account the parameter's scale. To address this issue, we use maximum value of each parameter to represent its scale and use the product of the coefficient in the maximum value of that particular parameter as the comparison metric for determining the *significance* of different parameters on the performance metrics.

It should also be noted that, in general, using highly correlated parameters as input for linear regression could mislead the algorithm and should be avoided. For example, using both *average upload rate* and *maximum upload rate* as predictor variables can lead to an unreliable model due to high correlation between them. Therefore, we include a shortened list of parameters representing different aspects of peer-level and group-level properties that we expect to be independent. We include *median upload rate* and *median contribution ratio* to represent peer properties and we add *average group population*, *average seed percentage*, *average content availability* and *average arrival rate* to represent peer-view of group properties in the regression input.

To evaluate the reliability and usability of the resulting model, *R-squared* value is reported for each generated model. This value

represents the prediction ability of the generated model (an R-squared value of 1 indicates a perfect prediction). For each coefficient, a *p-value* is provided [§] that can be used to determine the reliability of the coefficient.

Outliers can adversely affect a linear model. In this analysis, we try to discard the outliers from the model by cutting the top and bottom 5 percent of data with respect to different parameters. This technique slightly improved our model.

The following list provides the steps we have taken to both improve and simplify the model. The resulting models for each step are presented in Table 6.4 for the RedHat torrent. In all cases the outlier removal technique described above has been implemented.

Scenario 1: We use the following parameters as the input to model the performance metrics: The parameters are: outbw.50p, contrib.50p, avg.grp.pop, avg.grp.seed.perc, avg.grp.cont.avail, avg.grp.arr.rate. The resulting R-squared is 0.176 which does not indicate a good linear model.

Scenario 2: Some parameters used in the model have a large range making the corresponding model coefficients very small. Using logarithmic scale, we can limit their variations. Logarithmic scale can also improve the model where relation between the predictor and the predicted parameters is not linear. We change the larger scale variables to log scale (outgoing bandwidth and population) one by one improving the model slightly. In Scenario 2 we use replace $outbw.50p$ with $log(outbw.50p)$. This technique improves the R-square of the resulting model from 0.176 to 1.189 which is a slight improvement.

Scenario 3: From Scenario 1, replace $avg.grp.pop$ with $log(avg.grp.pop)$. This technique only improves the R-square of the resulting model by about 0.002.

Scenario 4: From Scenario 1, replace both $outbw.50p$ a nd $avg.grp.pop$ by $log(outbw.50p)$ and $log(avg.grp.pop)$ respectively. The resulting R-squared indicates 0.017 improvement comparing to the base case.

Scenario 5: S-plus provides a procedure to simplify the model by discarding non-important parameters. The method called *step()* tries different combinations of the supplied variables to gain the best and simplest model. Using this method we simplify the model. Applying step() to the model in Scenario 4, step() decides to remove $avg.grp.seed.perc$ from the input parameters list. This will only reduce the R-squared by 0.001.

Discussion:

Using linear model, we found that seed percentage is not an important factor in peer performance. Average group content availability seems to be the most important factor together with the peer's median uploading rate.

In general, the linear model is not a good estimator for either of the performance metrics. The value of multiple R-squared for the utilization metric is 0.19 and for the normalized standard deviation, it is only 0.0012. However the p-values calculated for the coefficients are generally less than 0.1 and in many cases less than 0.01 which shows that the coefficient value is very reliable.

6.6. Data Mining

As another analysis method, we use Weka's C4.5 decision tree generating algorithm. In this

[§] In statistical hypothesis testing, the p-value is the probability of obtaining a result at least as "impressive" as that obtained, assuming the truth of the null hypothesis that the finding was the result of chance alone. [13]

Perf. Metric	Test Mode	No. of Params	No. of Leaves	Correctly Classified Instances
Downlink Utilization	10-fold cross validation	29	1144	89%
Normalized St. Dev.	10-fold cross validation	29	877	93%

Table 7. Studied data-set information

method we divide the peers into 4 groups based on their performance metric. We choose the group boundaries to be 25th, 50th and 75th percentile of the performance metric. The tables used in this technique include all peer-level and group-level properties and the decision tree's goal is to predict which performance group each peer belongs to. In Table 6.6 the decision tree results are provided for both performance metrics for the RedHat torrent.

7. CONCLUSION AND FUTURE WORK

In this study we used BitTorrent tracker logs in a statistical analysis to examine BitTorrent's peer-level and group-level characteristics. We used Spearman's rank correlation and linear regression to discover possible relationships between peer and group properties with the observed performance by individual peers. In this study we found that: (i) BitTorrent performs well in accommodating flash crowds by turning new peers into seeds quickly. (ii) There is a good amount of altruism that together with incentives, keep enough resources almost always available in a torrent with a minimum number of participating peers.

(iii) The statistical correlation test and regression indicated that there is no single factor determining peer performance. (iv) According to the correlation test and regression results, Peer's outgoing bandwidth, average content availability and churn are important factors in peer's performance. (v) In the torrents we studied, the amount of seed percentage and content availability were above the required threshold and therefore, their variations did not affect peers' performance.

(vi) The behavior of the system in practice is rather complex and chaotic due to inherent dynamics in peer participation and content delivery as well as bandwidth heterogeneity and asymmetry.

Many shortcomings of our approach can be addressed by performing an active measurement in BitTorrent that includes contacting all peers periodically and capturing each individual peer's progress in short time intervals as well as studying the segment diversity in the network. Using peer-exchange also one can try to characterize the topology of a BitTorrent overlay.

Acknowledgment:

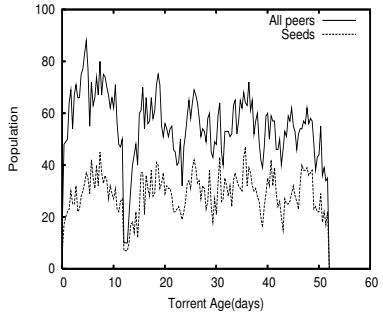
This work has only become possible with tremendous amount of help, support and encouragement from my advisor Prof. Reza Rejaie. I also thank Daniel Stutzbach for his insightful ideas and for providing me with some tools and data sets. I also benefited from discussions with Nick Duffield (AT&T Labs Research) and David Levin (Math Dept., University of Oregon) as well as helps from Prof. Dejing Dou and his student Daya Wimalasuriya for their help in data mining. I am grateful to my committee members Prof. Li and Prof. Proskurowski for their time and attention. Finally, special thanks go to my colleague, friend and wife Nazanin Magharei for her love, support and help.

This project was in part funded by NSF.

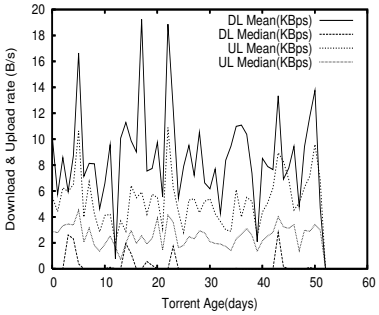
REFERENCES

1. A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *INFOCOM*, 2006.
2. B. Cohen. Incentives Build Robustness in BitTorrent. http://www.bittorrent.com/bittorrentecon.pdf, 2003.
3. D. Erman, D. Ilie, A. Popescu, and A. A. Nilsson. Measurement and Analysis of BitTorrent Signaling Traffic. In *Nordic Teletraffic Seminar*, 2004.
4. L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, Analysis, and Modeling of BitTorrent-like Systems. In *Internet Measurement Conference*, 2005.
5. M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, 2004.
6. T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *Internet Measurement Conference*, 2005.
7. MaxMind. Geo-iP. http://www.maxmind.com/app/ip-location, 2006.
8. A. Parker. P2P in 2005. http://www.cachelogic.com/research/2005_slide01.php, 2005.
9. J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-sharing System: Measurements and Analysis. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.
10. D. Qiu and R. Srikant. Modeling and Performance Analysis of Bit Torrent-Like Peer-to-Peer Networks. In *SIGCOMM*, 2004.
11. D. Stutzbach and R. Rejaie. Capturing Accurate Snapshots of the Gnutella Network. In *Global Internet Symposium*, 2005.
12. Y. Tian, D. Wu, and K.-W. Ng. Modeling, Analysis and Improvement for BitTorrent-Like File Sharing Networks. In *INFOCOM*, 2006.
13. Wikipedia. P-value. http://en.wikipedia.org/wiki/P-value, 2006.
14. Wikipedia. Spearman's rank correlation coefficient. http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient, 2006.
15. G. Wu and T. Chiueh. How efficient is BitTorrent? In *MMCN*, 2006.

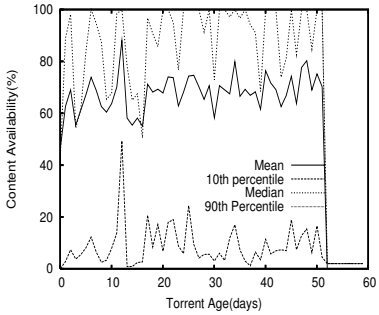
APPENDIX A. EVOLUTION OF GROUP PROPERTIES IN OTHER TORRENTS



(a) Population evolution in 2 months

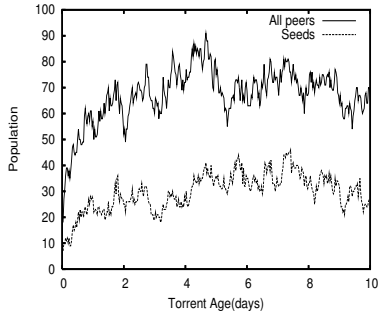


(b) Download and upload rates in 2 months

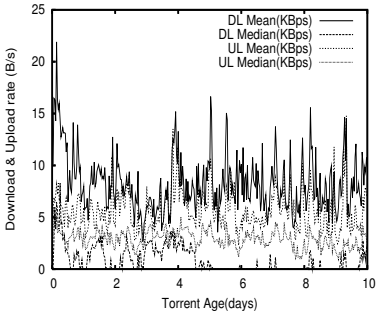


(c) Content availability in 2 months

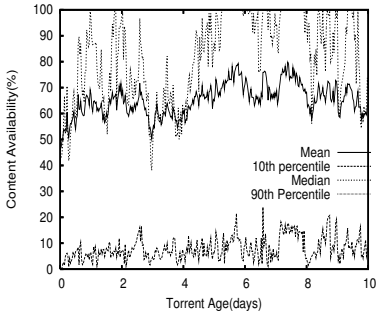
Figure 7. Evolution of group properties for the Debian torrent during 60 days using 2 samples per day (sampled view)



(a) Population in the first 10 days

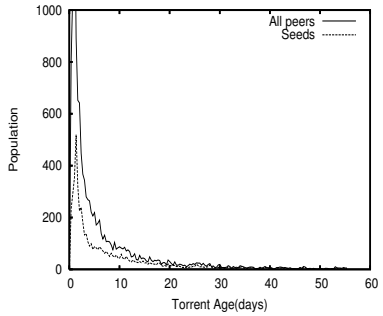


(b) Download and upload rates in the first 10 days

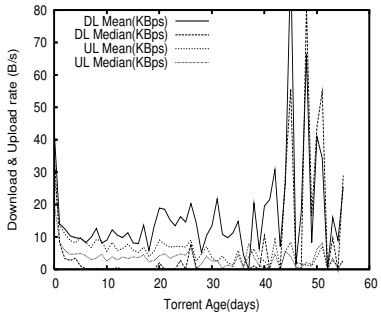


(c) Content availability in the first 10 days

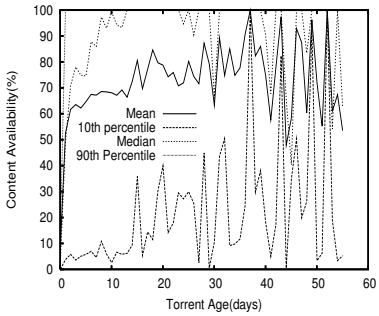
Figure 8. Evolution of group properties for the Debian torrent during the first 10 days using using one sample every 40 minutes (zoom-in view)



(a) Population evolution in 2 months

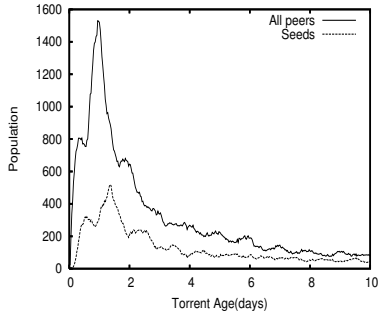


(b) Download and upload rates in 2 months

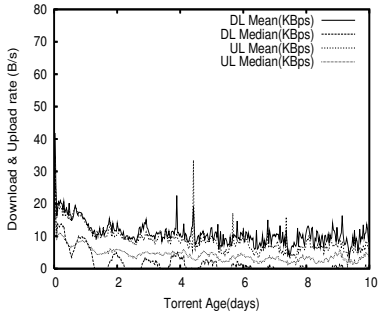


(c) Content availability in 2 months

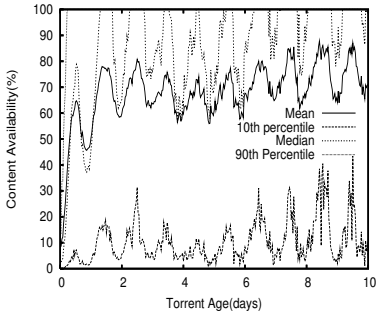
Figure 9. Evolution of group properties for the Gaming torrent during 60 days using 2 samples per day (sampled view)



(a) Population in the first 10 days



(b) Download and upload rates in the first 10 days



(c) Content availability in the first 10 days

Figure 10. Evolution of group properties for the Gaming torrent during the first 10 days using using one sample every 40 minutes (zoom-in view)