BitTorrent Session Characteristics and Models

David Erman Dragos Ilie Adrian Popescu

Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
371 79 Karlskrona, Sweden

Abstract

The paper reports on a modeling and evaluation study of session characteristics of BitTorrent traffic. BitTorrent is a second generation Peer-to-Peer (P2P) application recently developed as an alternative to the classical client-server model to reduce the load burden on content servers and networks. Results are reported on measuring, modeling and analysis of application layer traces collected at the Blekinge Institute of Technology (BIT) and a local ISP. For doing this, a dedicated measurement infrastructure has been developed at BIT to collect P2P traffic. A dedicated modeling methodology has been put forth as well. New results are reported on session characteristics of BitTorrent, and it is observed that session interarrivals can be accurately modeled by the hyper-exponential distribution while session durations and sizes can be reasonably well modeled by the lognormal distribution.

Keywords: BitTorrent, traffic measurements, traffic modeling, traffic self-similarity

1 Introduction

Over the last years, P2P file sharing systems have evolved to be some of the major traffic contributors in the Internet [19]. Although an exact definition of "P2P systems" is still debatable, such a system typically represents a distributed computing paradigm where a spontaneous, continuously changing group of collaborating computers act as equals in supporting applications such as resource redundancy, content distribution, and other collaborative actions. The roles of the computers is determined based on the perceived system performance.

There are currently a number of architectural designs for P2P systems, which follow different strategies used for resource discovery and content distribution [7, 18, 22]. For instance, resource discovery can be done either with the help of a centralized directory (e.g., Napster [23]) or with the help of a decentralized directory (e.g., KaZaa/FastTrack [24]) or with the help of query flooding (e.g., Gnutella [20]). On the other hand, content distribution can be done either in a distributed way (so-called "pure P2P" system, e.g., Gnutella) or server mediated (so-called "hybrid P2P", e.g., Napster, BitTorrent [4]) or based on a hybrid client/server model (e.g., SETI [26], GRID [11]) or even based on a pure client/server model (e.g., WWW). Specific advantages and drawbacks are associated with every architectural design, as reported in [18, 22].

There are several important consequences related to the appearance of P2P systems. An important one is related to the high traffic volumes caused by P2P systems, which are due to both signaling traffic and data traffic. Furthermore, another serious consequence is related to the high variability introduced by P2P systems in the Internet traffic patterns, with fluctuations that strongly variate in time and space. For instance, recent measurement studies showed that P2P traffic may come up to 80% of the total traffic in a high speed IP backbone link carrying TCP traffic towards several ADSL areas and that Long Range Dependence (LRD) properties and the degree of traffic self-similarity in traffic seem to reduce with the predominance of P2P traffic [1]. Other open problems in P2P systems are related to the appearance of "mice" (short transfers) and "elephants" (long transfers) phenomenon in Internet traffic, scalability, expressiveness, efficiency and robustness of search mechanisms as well as security issues.

Measurement studies and analysis of P2P traffic have been rather limited so far. This is because of the complexity of this task, which involves answering hard questions related to data retrieval and content location, storage, data analysis and modeling of traffical and topological characteristics as well as privacy and copyright issues. Furthermore, the appearance of what we call the second generation P2P protocols, best exemplified by BitTorrent [12], further complicated the picture. Compared to the first generation P2P protocols, BitTorrent relies on swarming techniques in combination with the "tit-for-tat" mechanism for creating incentive in content distribution. No search functionality is built into the protocol, and the signaling is geared towards an efficient dissemination of data [10]. BitTorrent has become extremely popular over the last years. According to Cachelogic, the BitTorrent traffic volume has increased from 26% to 52% of the total P2P traffic volume during the first half of 2004 [3]. The first generation P2P protocols (e.g., Gnutella) are intensively using signaling traffic as well as the exchange of user resources [13]. This protocol diversity further challenges the task of P2P traffic measurements and analysis.

There are actually not so many measurement studies done on BitTorrent [14, 6, 15]. This is because the protocol is quite new, only a few years old. In these studies, traffic has been collected from "tracker" as well as with the help of modified clients. The drawback in using modified clients is related to the accuracy of the timestamps at the application level, which is directly depending on the type and version of the computer hardware, OS and application software.

The main goals of the paper are towards an understanding of the characteristics of BitTorrent sessions, to be further used in a P2P simulation environment. For doing that, we have designed a dedicated measurement system to do traffic

measurements on P2P systems [12]. Detailed results are reported on measuring, modeling and analysis of BitTorrent traffic collected from BIT, Karlskrona as well as at a local ISP with 5 Mbps link. Our results show that BitTorrent session interarrivals can be accurately modeled by the hyper-exponential distribution while session durations and sizes can be reasonably well modeled by the lognormal distribution.

The rest of the paper is organized as follows. In Section 2 we provide a short overview of the BitTorrent protocol. In Section 3 we describe the P2P measurement infrastructure developed at BIT. Section 4 reports on the BitTorrent traffic measurements done at BIT and a local ISP. Section 5 presents the traffic metrics used in the evaluation of BitTorrent. In Section 6 we describe the modeling methodology used in our experiments. Section 7 describes the BitTorrent session characteristics with summary statistics. Finally, Section 8 concludes the paper.

2 The BitTorrent Protocol

BitTorrent is a P2P protocol for content distribution and replication designed to quickly, efficiently and fairly replicate data [4]. In contrast to other P2P protocols, the BitTorrent protocol does not provide any resource query or lookup functionality, but rather focuses on fair and effective replication and distribution of data. The signaling is geared towards an efficient dissemination of data only. The protocol is fair in the sense that peers exchange content in a tit-for-tat fashion. Non-uploading peers are only sporadically allowed to download. The protocol operates over TCP and uses swarming, i.e., peers are downloading parts, the so-called *pieces*, of the content from several peers simultaneously. The consequence of this is efficient network utilization. The size of the pieces is fixed on a per-resource basis and can not be changed.

A peer interested in downloading some content by using BitTorrent must first obtain a set of metadata, the so-called *torrent* file, to be able to join a set of peers engaging in the distribution of the specific content. In the following we use the term *swarm*, or *distribution swarm*, to define a set of metadata together with the associated network entities. The metadata needed to join a BitTorrent swarm consists of the network address information (in BitTorrent terminology called the announce URL) of the tracker and resource information such as file size and piece size. An important part of the resource information is a set of Secure Hash Algorithm One (SHA-1) hash values, each corresponding to a specific piece of the resource. These hash values are used to verify the correct reception of a piece. The resource information is also used to calculate a separate SHA-1 hash value, the *info* field, used as an identification of the current swarm. The hash value appears in both the tracker and peer protocols. The metadata does not contain any information regarding the peers participating in a swarm.

A BitTorrent distribution swarm can be partitioned into three network entities and two protocols. The first network entity is a centralized software entity, the so-called *tracker*, which keeps lists of connected peers as well as information about their evolution. The tracker replies to peer requests for other peer addresses and ports as well as records simple statistics about the evolution of the swarm. The second entity is the set of active peers, which can be further divided into *seeds* and downloading peers, or *leechers*. A seed is defined to be a peer that has already retrieved an entire file or amount of data, and has stopped downloading data from other peers. A seed however may continue to serve other peers. Also, an initial seed is necessary for peers to be able to start replicating the content. Finally, the third network entity is a server, usually a webserver, which provides the metadata required for joining a specific swarm. The distribution of the metadata is not necessarily done via HTTP, but it can be done in any manner. Any way of distributing the torrent file is valid.

The BitTorrent protocols (except the metadata distribution protocol) are the tracker protocol and the peer wire protocol. The tracker protocol uses HTTP. Peers make HTTP GET requests and the tracker sends responses in the returning HTTP response data. The purpose of the peer request to the tracker is to locate other peers in the distribution swarm and to allow the tracker to record simple swarm statistics. The peer sends a request containing information about itself and some basic statistics to the tracker, which responds with a randomly selected subset of all peers engaged in the swarm.

The peer wire protocol operates over TCP, and uses in-band signaling for peer communication. Signaling and data transfer are done in the form of a continuous bi-directional stream of fixed-size, length-prefixed protocol messages. A P2P session is equivalent with a TCP session, and there are no protocol entities for tearing down a BitTorrent session beyond the TCP teardown itself. Connections between peers are single TCP sessions, carrying both data and signaling traffic. Once a TCP connection between two peers is established, the initiating peer sends a handshake message containing the peer id and info field hash (Figure 1). If the receiving peer replies with the corresponding information, the BitTorrent session is considered to be opened and the peers start exchanging messages across the TCP streams. In other cases, the TCP connection is closed. Immediately following the handshake procedure, each peer sends information about the pieces of the resource it possesses. This is done only once, and only by using the first message after the handshake. The information is sent in a *bitfield* message, consisting of a stream of bits, with each bit index corresponding to a piece index.

A peer maintains two states for each peer relationship, namely *interested* and *choked*. If a peer is choked, then it will not receive any data unless unchoking occurs. Usually, unchoking is equivalent with uploading. The *interested* state indicates whether other peers have parts of the sought content. Interest should be expressed explicitly, as should lack of interest. That means that a peer wishing to download notifies the sending peer (where the sought data is) by sending an *interested* message, and as soon as the peer no longer needs any other data, a *not interested* message is issued. Similarly, for a peer to be allowed to download, it must have received an *unchoke* message from the sending peer. Once a peer receives a *choke* message, it will no longer be allowed to download. This allows the sending peer to keep track of the peers that start downloading when unchoked. A new connection starts out choked and not interested. A peer with all data, i.e., a seed, is never interested.

The choke/unchoke and interested/not interested mechanism provides fairness in the BitTorrent protocol. As it is the transmitting peer that decides whether to allow a download or not, peers not sharing content will be reciprocated in the same manner. To allow peers that have no content to join the swarm and start sharing, a mechanism called *optimistic unchoking* is employed. From time to time, a peer with content will allow even a non-sharing peer to download.

Data transfer is done in parts of a *piece* (called *sub-piece*) at a time, by issuing a *request* message. The sub-pieces are typically of size 16384 or 32768 bytes. To allow TCP to increase the throughput, several requests are usually sent back-to-back. Each request should result in the corresponding sub-piece to be transmitted. If the sub-piece is not received within a certain time (typically one minute), the non-transmitting peer is snubbed, i.e., it is punished by not being allowed to download, even if unchoked. Data transfer is done by sending a piece message, which contains the requested sub-piece

(Figure 2). Once the entire piece, i.e., all sub-pieces, has been received, and the SHA-1 hash of the piece has been verified to be correct, a *have* message is sent to all connected peers.

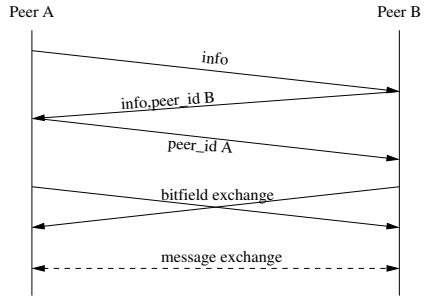


Figure 1: BitTorrent handshake procedure

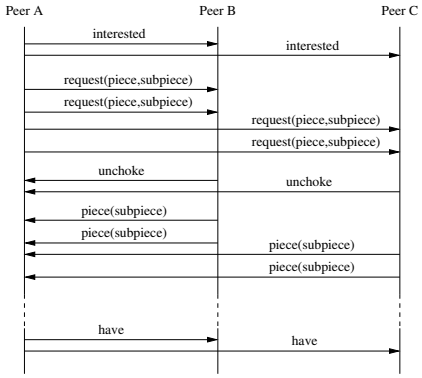


Figure 2: BitTorrent protocol exchange

3 Traffic Measurements

A mixed methodology for traffic measurements of P2P systems has been developed at BIT, which is based on a combination of instrumentation at the application layer with transport flow identification and extraction of packets captured at the link-layer (Fig. 3). This solution allows accurate measurements on both generations of P2P protocols.

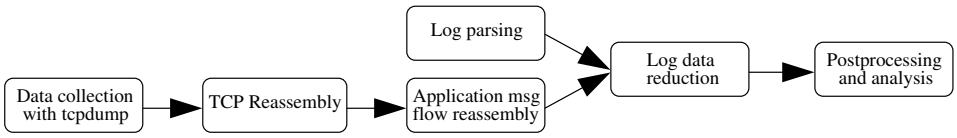


Figure 3: Measurement procedures

The P2P measurement infrastructure developed at BIT consists of peer nodes and protocol decoding software [9]. `Tcpdump` [16] and `tcptrace` [25] are used for traffic recording and protocol decoding. Although the infrastructure is currently geared towards P2P protocols, it can be easily extended to measure other protocols running over TCP as well. Furthermore, we plan to develop similar modules to measure UDP-based applications as well.

The BIT measurement nodes run the Gentoo Linux 1.4 operating system, with kernel version 2.6.5. Each node is equipped with an Intel Celeron 2.4GHz processor, 1GB RAM, 120GB hard drive, and 10/100 FastEthernet network interface. As shown in Fig. 4, the network interface is connected to a 100Mbit switch in the lab at our department, which is further connected through a router to the GigaSUNET backbone.

Our experience with the current setup has been that the traffic recording step alone accounts for about 70% of the total time taken by measurements. Protocol decoding is not possible when the hosts are recording traffic. The main reason is the protocol decoding phase, which is I/O intensive and requires large amounts of CPU power and RAM memory. To overcome this problem we are developing the distributed measurement infrastructure shown in Fig. 5.

When used in the distributed infrastructure the P2P nodes are equipped with an additional network interface, which we refer to as the *management* interface. P2P traffic is recorded from the primary interface and stored in a directory on the disk. The directory is exported using the Network File System (NFS) over the management interface. Data processing workstations can read recorded data over NFS as soon as it is available. Optionally, the data processing workstations can be located in private LAN or VPN in order to increase security, save IP address space and decrease the number of collisions on the Ethernet segment. In this case, the Internet access router provides Internet access to the workstations, if needed.

4 BitTorrent Measurements

Two sets of BitTorrent measurements have been performed. The first set used the instrumented version of the reference BitTorrent client as the main measurement tool, with only partial packet capture to determine timestamp accuracy. The second set involved full packet capture and stream reassembly in addition to application logging.

The traffic for the first set of measurements was collected at two different locations over a three-week time period starting on May 3rd 2004. One location was the networking lab at BIT (100 Mbps Ethernet) and the other one was a local ISP with a 5 Mbps link. The measurements represent 12 different runs (with lengths of 2 to 12 days) of the instrumented client, 3 of which were run as the only active application. This was done so as to establish a point of reference without applications competing for available bandwidth. To measure more realistic scenarios, the rest of the runs were done with some temporal overlap [10]. A total of 20GB of uncompressed XML logs were collected in the first set of measurements. After postprocessing, the amount of logs was over 25GB. The logs contain approximately 100 million protocol messages from almost 300000 individual sessions. The BitTorrent log files contain a list of client software states, e.g., tracker announcements, new connections, choke, unchoke, interested, uninterested, along with the timestamps when the state change took place.

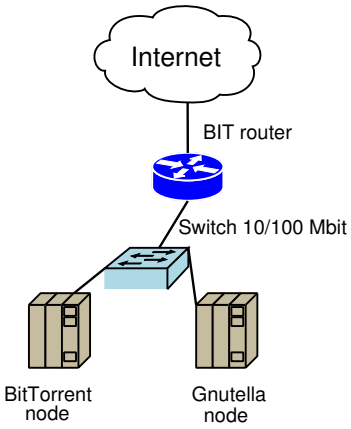


Figure 4: Measurement setup

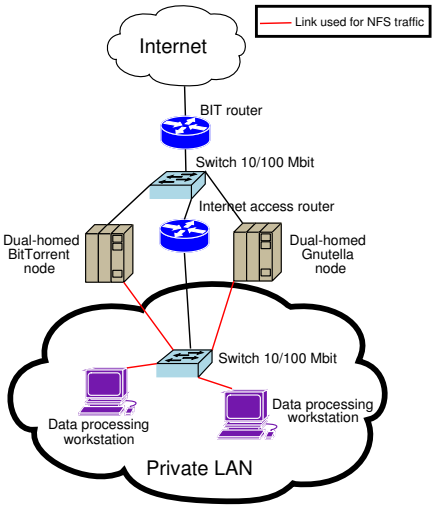


Figure 5: Distributed measurement setup

The second set of traces were collected as `tcpdump` traces at the BIT networking lab during one week, starting June 4th, 2004. A single instance of the reference client was run as the only application on the measurement node. The set contains 150GB of data, out of which 143GB are `tcpdump` traces. The rest of the data are application logs and postprocessed logs. Approximately 22 million messages were transmitted in 53000 sessions during the second measurement set.

An important issue regarding traffic measurements in P2P networks is the copyright issue. The most popular content in these networks is often copyrighted material. To circumvent this problem, we joined BitTorrent swarms distributing several popular Linux operating system distributions.

5 Traffic Metrics

The BitTorrent client application logs are in essence timestamped protocol events. This means that metrics like interarrival and interdeparture times are readily available by simple calculations. The possibility does exist to compute detailed statistics on several levels of aggregation as well. Most notably, this offers the possibility to look into potential burstiness on timescales that are decided by the timestamp accuracy.

Out of substantial amounts of logged data, specific software has been written to extract several important statistics and metrics, to characterize the peer behavior only, and not the entire swarm [9]. To measure the true size of the swarm, active probing of the tracker is necessary. This is, however, subject for future work. The goal is to use accurate characterization and modeling of the behavior of a peer in modeling entire swarms.

A number of metrics have been used for the characterization of the BitTorrent signaling traffic [9]. The most important ones are as follows:

Download time
This is the time it takes for the modified client to do a complete download. This metric also provides information about the peer changes from being both a downloading and uploading peer to being a seed, thus offering the possibility to collect statistics about the seed and leecher states.

Session duration and size
A BitTorrent session is equivalent with a TCP session, given that the BitTorrent handshake is completed. As BitTorrent protocol messages are fixed-length messages, there is a one-to-one mapping between the messages sent and received during a session and the session size. A BitTorrent session time is given by the TCP session time, whereas the session size is given by the amount of data transmitted during the TCP session.

Number and type of messages
We count the number of messages of each type in both upstream and downstream directions. Together with the session duration and size, this gives us valuable insights into the behavior of a peer.

Host persistence
We also count the number of unique host IP addresses and peer client IDs. If a given host IP address has a one-to-one mapping to a peer ID and we have a long session time, the peer is considered to be persistent. Persistent peers indicate a healthy swarm in the sense that new peers are more likely to find a larger number of seeds in a swarm with many persistent peers than in one with less persistent peers.

Peer swarm size
The peer swarm size refers to the number of peers observed by the measuring client at any given time. This is not the size of the entire swarm, i.e., the total number of collaborating peers, but the number of peers to which the measuring peer is connected. Information about the total swarm size is only available at the tracker, and therefore it is not considered in the reported measurements.

Piece response times
The piece response time is defined to be the time elapsed between the moment of the initial *request* for any subpiece belonging to a given piece to the moment of the transmission of the associated *have* message. This parameter gives us the possibility to estimate the downstream bandwidth usage.

Piece popularity
The popularity of a piece is given by the number of requests for any subpiece of a given piece. This gives an indication of the effectiveness of the piece selection algorithms of the requesting peers.

6 Modeling Methodology

Detection and estimation of heavy-tailed properties in the distribution of application layer objects is an important part in performance modeling of applications. It may for instance reveal the presence of infinite mean or variance. Accurate estimation of these properties is also important in order to capture the degree of Long-Range Dependence (LRD) inherent in the objects. Such estimates are also useful in building simulation models that can reproduce traffic conditions as observed in real networks.

Often the random variable possessing heavy tail appears hidden behind another distribution. While the two distributions may have very different tail behavior in a mathematical sense, it may be quite difficult to segregate the two in a practical fitting problem. The crux of the problem lies therefore in determining the cutoff point between the two distributions [27, 5, 17].

The modeling process for mixture models is partitioned into three separate activities: distribution selection, parameter estimation and fitness assessment.

6.1 Distribution selection

The first step is to do a visual inspection of various plots such as histogram (or experimental probability density function (PDF)), empirical distribution function (EDF), complementary cumulative distribution function (CCDF), Hill plots and α-estimation plots [5]. We inspect the lower quantiles of the data using the PDF and CCDF for the upper tail. The CCDF is useful for discerning potentially heavy tail behavior in the distribution such as for file sizes and session durations [21]. The histogram is more suitable for observing metrics in situations where higher frequency behavior is to be modeled, such as for interarrival times. Hill plots give an indication of the amount of heavy tail behavior, and also potential cutoff points in the mixture model case. The α-estimation provides indications of the degree of self-similarity in the data.

The visual inspection helps in eliminating many candidate distributions, and indicates whether a single distribution will suffice or if a mixture model is required. For this work, we primarily consider single distributions and mixtures of two distributions, as the number of measurements makes the heuristics involved in calculating more cutoff points prohibitively complex.

6.2 Parameter estimation

Based on the candidate distributions selected for modeling, we employ Maximum Likelihood Estimation (MLE) to obtain parameter estimates. With the number of sessions available for the measurements, we assume that the parameter estimations obtained are accurate enough to consider the associated distribution fully specified, given that the confidence intervals for the estimated parameters are within acceptable boundaries.

In the case of single distributions, the parameter estimation is a straightforward procedure, and estimates are obtained from the complete set of data. In the mixture model case, we use successive right censoring as employed in [17] together with an error percentage assessment (described in the following section) to find out the cutoff points for the mixture model.

6.3 Fitness assessment

To determine whether a distribution is representative of the observed data, we employ visual procedures, formal hypothesis tests, and an error percentage assessment. We use visual procedures like histogram, CCDF overplots and quantile-quantile (QQ) plots. Overplots give insight in the fitness of the lower and upper tails respectively of a single distribution. We use the QQ plot as a visual aid to assess the representativeness of the chosen model to several measurements simultaneously.

Formally defined goodness-of-fit hypothesis tests such as the Kolmogorov-Smirnov (KS), λ^2 and Anderson-Darling (AD) tests are used to test the null hypothesis H_0 : "*The samples* $X_1 \dots X_n$ *are drawn from a distribution* $F(x;\Theta)$" [8]. There is however a major drawback related with these types of tests namely that they always tend to reject the null hypothesis in the case of large sample [2]. A possible reason could be the parametrization errors, even if these errors are ever so slight. This is especially true for EDF tests that need modified test statistics, which depend on the size of the data set, e.g., the KS and Cramér-von Mises tests. We use the Anderson-Darling statistic as an additional goodness-of-fit measure for the metrics where the number of samples is relatively low. For larger sample sets however we use a different method, as described below.

To assess the quality of the fitted distributions in a more quantitative manner, we employ a method similar to the EDF test but that does not suffer as much with increasing size of sample space. For the case of single distribution, the fitness assessment is the final step of the modeling, as we accept the MLE estimated parameters and do not perform any further parameter optimisation. On the other hand, in the case of mixture model, we use this step as part of the process of locating a suitable cutoff point between the distributions making up the mixture.

The method is based on the EDF test for a fully specified distribution, as described in [8]:

1. Obtain the order statistics $X_1 < X_2 < \cdots < X_n$ from the measured data.

2. Transform the original data by using the probability integral transform (PIT) method and using the selected distribution and estimated parameters. If the samples $X_1 \cdots X_n$ are IID samples from some distribution F, then $\hat{U}_i = F(X_i; \hat{\Theta})$,where $i = 1, 2 \ldots n$, are uniformly IID on $[0, 1]$.

3. Obtain the error percentage by using the following expression:

$$E_{\%} = 100 \times \frac{\sum_{i=1}^{n} |U_i - \hat{U}_i|}{nE_{max}} \tag{1}$$

where E_{max} is defined as $\int_0^1 \sup\{U(x), 1 - U(x)\}\, dx = 0.75$ or, in plain terms, the maximum discrepancy from a true $U[0, 1]$ distribution that may occur.

4. Accept or discard the fitted distribution as "good enough" according to some predefined criteria. In our case, we choose $E_{\%} \approx 5$ as an upper limit for not discarding the fitted match. It is important to mention that this is not a statistical significance level, but rather an acceptable margin of error.

   Additionally, fuzzy classification or rough set theory may be employed in quantifying the goodness-of-fit in a more formal way. We use the informal degrees of fitting quality presented in Table 1. More formally defined measures, e.g., proper membership functions, are subject of future research.

$E_{\%} \approx$	0	1	2	3	4
Degree	perfect	very good	good	fair	poor

Table 1: Fitness quality boundaries

7 Session characteristics

In this section we report the modeling results for the distributions of session interarrival times, upstream session sizes and durations. Table 2 provides a summary of the number of sessions in each of the thirteen measurements, except for number 9, which was lost due to hardware failure. It is observed that measurement 6 is different, with regards to both mean session size and session duration. Further, the maximum session size for this measurement is more than twice that of any other measurement. The mean session size is observed to be about twice that of the corresponding measurement of the same content (measurement 10). As measurements 6 and 10 have the largest session sizes, it is very likely that the session size is related in this case to the total content size (4.3GB).

The minimum session durations are all set to 0, indicating that all of them are shorter than the accuracy provided for by the application logs. These very short sessions are also indicated in the minimum session sizes, and they correspond to a session containing only a handshake or an interrupted handshake. More detailed information is available in [9].

Measurement number	Sessions	Session duration (s)				Session size (MB)			
		Mean	Max	Min	Std	Mean	Max	Min[a]	Std
1	29712	343	98991	0	2741	27.49	647.26	73	70.65
2	46022	233	117605	0	2316	27.15	646.03	73	64.05
3	28687	465	171074	0	3614	28.54	539.20	73	61.70
4	13493	750	143707	0	3942	49.88	671.99	73	100.65
5	12354	910	180298	0	4504	57.08	668.53	73	116.10
6	10685	1207	223235	0	7016	74.25	3117.79	73	247.74
7	4444	218	46478	0	1642	49.96	431.13	78	76.48
8	17287	231	87026	0	1972	33.11	695.94	73	109.31
10	9701	652	267497	0	5907	37.78	1499.85	73	109.08
11	43939	448	141509	0	3791	17.22	475.86	73	52.73
12	68288	197	292241	0	2580	8.31	987.89	73	30.63
13	52833	465	483996	0	4036	32.2	1652.83	73	99.4

Table 2: Session and peer summary

[a]This column measured in bytes.

7.1 Session interarrivals

The reported distributions refer to interarrival times for remotely initiated sessions during the seeding phase of our measurement peer. We do not consider the leech phase, partly because it is short compared to the seed phase and the number of non-locally initiated sessions is fairly low, and partly because the peer is more active during this phase than during the seed phase. The combination of active peer status and low number of samples (e.g., only 10–20 sessions) that is present during the leech phase makes the analysis more difficult.

We have modeled the session interarrivals by using a two-stage hyperexponential distribution, denoted by H_2. The associated probability density function is

$$H_2(x) = p\lambda_1 e^{-\lambda_1 x} + (1 - p)\lambda_2 e^{-\lambda_2 x} \tag{2}$$

where λ_1 and λ_2 are the arrival rates for the two exponentials, and p is the probability of an arrival being drawn from the first exponential term. In Figure 6 we present examples of visual assessment tools. Figures 6(a) and 6(b) show PDF and CCDF overlay plots for measurement 3. Both indicate a very good fitting for up to 99% probability mass, with most of the errors in the tail of the distribution. Figure 6(c) shows a QQ plot with all measurements.

Parameter estimates for each of the measurements have been obtained by using a maximum likelihood estimator. Table 3 reports the parameter estimates and the associated standard deviations obtained in the fitting procedure. Also presented is the $E_{\%}$ value and the resulting fitness decision and degree.

Measurement number	$\hat{\lambda}_1 \pm \hat{\sigma}_{\lambda_1}$	$\hat{\lambda}_2 \pm \hat{\sigma}_{\lambda_2}$	$\hat{p} \pm \hat{\sigma}_p$	$E_{\%}$	Comment
1	0.0593 ± 0.0046	0.1696 ± 0.0085	0.2215 ± 0.0467	2.07367	Pass, fair
2	0.1158 ± 0.0009	0.7556 ± 0.0279	0.7936 ± 0.0066	0.41535	Pass, very good
3	0.0566 ± 0.0006	0.3653 ± 0.0099	0.6575 ± 0.0077	0.49009	Pass, very good
4	0.5372 ± 0.0178	0.0168 ± 0.0002	0.2533 ± 0.0052	2.79455	Pass, fair
5	0.5538 ± 0.0212	0.0162 ± 0.0002	0.2156 ± 0.0052	2.79722	Pass, fair
6	0.4798 ± 0.0174	0.0127 ± 0.0002	0.2879 ± 0.0060	3.93588	Pass, poor
7	0.4188 ± 0.0143	0.0052 ± 0.0001	0.3014 ± 0.0076	2.05430	Pass, good
8	0.5142 ± 0.0113	0.0168 ± 0.0002	0.4252 ± 0.0050	2.79291	Pass, fair
10	0.5581 ± 0.0205	0.0128 ± 0.0002	0.3276 ± 0.0064	3.76412	Pass, poor
11	0.0140 ± 0.0009	0.0802 ± 0.0005	0.0219 ± 0.0024	2.20763	Pass, good
12	0.0935 ± 0.0004	5.8224 ± 0.1380	0.8252 ± 0.0021	3.84606	Pass, poor
13	0.0563 ± 0.0004	0.4175 ± 0.0065	0.5897 ± 0.0048	1.87389	Pass, good

Table 3: Fitted hyperexponential parameters

Summarizing the results for session interarrivals during the seeding phase we observe that all measurements pass according to the selected error criteria. Furthermore it is observed that measurements 2 and 3 have low $E_{\%}$ values, and they show significance levels of $\approx$0.005 when using the Anderson-Darling test. This is an indication for good quality of fitting for the selected distributions.

7.2 Session duration and size

In this section we report the modeling results for the size and duration of remotely initiated peer sessions. We observe that they are highly related, and also show fairly high correlation, as shown in Table 4.

Measurement	1	2	3	4	5	6	7	8	10	11	12	13
ρ_{xy}	0.32	0.36	0.29	0.30	0.30	0.34	0.47	0.40	0.67	0.43	0.38	0.25

Table 4: Correlation coefficients for session duration and sizes

For reasons similar to those considered at session interarrival times, we consider for modeling the following:

- Measurements with more than 20000 sessions
- Sessions that have been initiated after the start of the seeding phase
- Sessions that actually request and receive at least one piece.

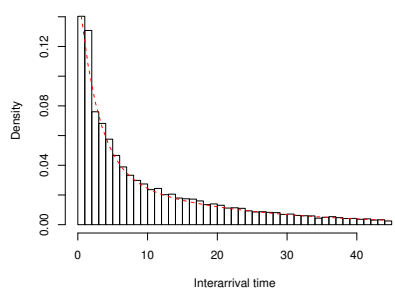
The reason for this is threefold:

- As observed in Table 5, most sessions do not transfer any data after the initial TCP handshake, with the consequence of a fairly low number of samples (3–6% of the total number of sessions) left for parameter estimation. By including the measurements with fewer sessions, the remaining number of sessions would be inadequate for proper parameter estimations.

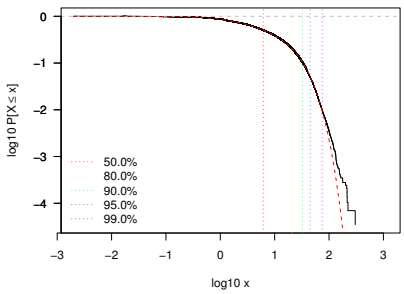
	Measurement	1	2	3	11	12	13
> 0 bytes	Sessions	1558	1619	1795	3092	3793	3438
	% of sessions	5	4	6	7	6	7
≥ 1 pieces	Sessions	1392	1356	1564	1769	2612	3017
	% of sessions	5	3	5	4	4	6

Table 5: Percentages of session sizes exceeding 0 bytes and 1 piece size

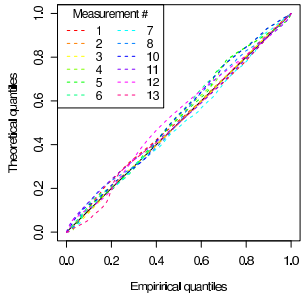
- The α-estimations for measurements (Table 6) indicate that there could be some heavy tail behavior present in the distributions, as observed in the CCDF plots. The shape in Figure 7(b) is representative for the CCDFs of session duration for all measurements. We observe clear multi-modal behavior, which means that the heuristic approach of locating the cutoff points, as described in [17], should be used.
- Both session sizes and durations appear to be drawn from a single, similar distribution when inspecting only sessions that have transmitted at least one piece (Figure 7).



(a) Empirical PDF for measurement 3 with fitted estimates overlaid



(b) CCDF for measurement 3 with fitted estimates overlaid

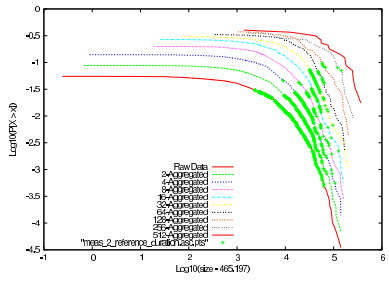


(c) QQ-plot of all measurements subject to $H_2(\hat{\lambda}_1, \hat{\lambda}_2, \hat{p})$

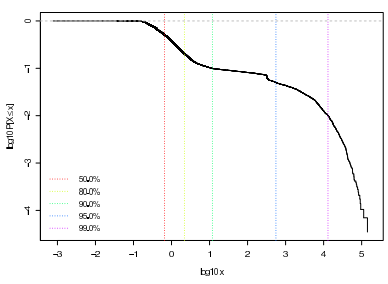
Figure 6: Fitness assessment plots

Measurement		1	2	3	11	12	13
duration	$\hat{\alpha}$	1.335	1.264	1.523	1.379	1.272	1.435
	$\hat{\sigma}^2_\alpha$	0.149	0.163	0.116	0.134	0.060	0.176
size	$\hat{\alpha}$	1.176	1.147	1.233	0.961	0.902	1.289
	$\hat{\sigma}^2_\alpha$	0.353	0.339	0.320	0.222	0.147	0.207

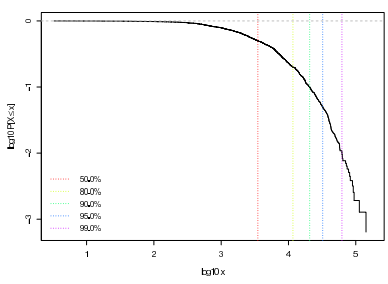
Table 6: Session α-estimates



(a) α-estimate plot for session duration



(b) Duration CCDF for all sessions



(c) Duration CCDF for sessions with ≥ 1 piece

Figure 7: α-estimates and CCDF for measurement 3

The models for session size and durations are reported in Tables 7 and 8, respectively. Only the sessions that actually receive data have been modeled. Lognormal distributions with parameters μ and σ have been used for modeling. The second and third columns show the estimated parameters, together with the associated estimated standard deviations, for which the best value of $E_{\%}$ was obtained. The value of $E_{\%}$ is given in column 6. The fourth column indicates the tail probability mass for which the fitting passed the 5% fitness limit of $E_{\%}$, while the fifth column shows the tail probability mass for which the *best* value was obtained.

Since the number of samples is substantially smaller than for the hyper-exponential models shown in section 7.1, we also calculate the Anderson-Darling statistic for the fitted distribution. Column 7 shows the significance levels obtained in the Anderson-Darling test, under the assumption that the parameter estimates are good enough to assume a fully specified distribution. The last column shows the fitting decision, together with the result of the AD test passing at the critical level.

Measurement number	$\hat{\mu} \pm \hat{\sigma}$	$\hat{\sigma}_{LN} \pm \hat{\sigma}$	Pass	Tail mass	$E_{\%}$	AD sign.	Comment
1	18.7 ± 0.04	0.62 ± 0.02	0.45	0.21	2.1	> 0.25	Pass, good; AD: Pass
2	17.8 ± 0.04	0.99 ± 0.03	1	0.4	2.9	> 0.025	Pass, fair; AD: Fail
3	18.4 ± 0.04	0.60 ± 0.02	1	0.24	3.3	> 0.05	Pass, fair; AD: Pass
11	14.1 ± 0.06	2.44 ± 0.04	1	0.99	2.4	≈ 0.001	Pass, good; AD: Fail
12	13.6 ± 0.05	2.36 ± 0.04	0.86	0.74	3.4	< 0.001	Pass, fair; AD: Fail
13	19.0 ± 0.03	0.69 ± 0.02	1	0.17	3.0	> 0.025	Pass, fair; AD: Fail

Table 7: Upstream size parameters

Measurement number	$\hat{\mu} \pm \hat{\sigma}$	$\hat{\sigma}_{LN} \pm \hat{\sigma}$	Pass	Tail mass	$E_{\%}$	AD sign.	Comment
1	8.55 ± 0.03	1.08 ± 0.02	1	0.74	2.2	≈ 0.01	Pass, good; AD: Fail
2	8.16 ± 0.04	1.33 ± 0.03	1	0.99	1.5	> 0.15	Pass, good; AD: Pass
3	8.17 ± 0.04	1.38 ± 0.02	1	0.98	1.6	> 0.05	Pass, good; AD: Pass
11	8.09 ± 0.04	1.56 ± 0.03	1	1	2.4	> 0.001	Pass, good; AD: Fail
12	7.2 ± 0.03	1.57 ± 0.02	1	1	3.9	$\ll 0.001$	Pass, poor; AD: Fail
13	7.94 ± 0.03	1.52 ± 0.02	1	1	2.3	< 0.001	Pass, good; AD: Fail

Table 8: Duration parameters

Though expected that a Pareto distribution or a mixture of the Pareto and log-normal distributions would provide a better fitting model, we found that this was not the case. We believe this is due to the limitation in the amount of data available in a BitTorrent swarm. There is no point in a peer downloading more data once the entire content is obtained.

For sessions not included in the model, i.e., sessions that do not receive any content, we have observed that session durations and sizes can be fairly accurately modeled by the inverted gamma distribution [9].

8 Conclusions

A characterization study of BitTorrent application traffic collected at two locations has been reported. Detailed results have been reported on measuring, modeling and analysis of the traffic collected. The modeling activity focuses on typical characteristics of remotely initiated peer sessions during the seeding phase of the measurement peer. New results have been reported on modeling BitTorrent session interarrival times, sizes and durations. Session interarrivals have been observed to be accurately modeled by the hyper-exponential distribution while session durations and sizes have been observed to be reasonably well modeled by the lognormal distribution.

Our future work will be on further modeling and analysis of the collected traces, to model the BitTorrent traffic at the message level as well as other relevant parameters like peer swarm size and piece popularity, to be further used in a P2P simulation environment.

References

[1] Nadia Ben Azzouna and Fabrice Guillemin. Experimental analysis of the impact of peer-to-peer applications on traffic in commercial ip networks. *European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services*, 2004.

[2] Jan Beran. *Statistics for Long-Memory Processes.* Chapman & Hall, 1994.

[3] Cachelogic. The true picture of peer-to-peer file sharing. http://www.cachelogic.com/research/slide9.php, May 2005.

[4] Bram Cohen. BitTorrent protocol specification. http://www.bitconjurer.org/BitTorrent/protocol.html, February 2005.

[5] Mark E. Crovella and Murad S. Taqqu. Estimating the heavy tail index from scaling properties. *Methodology and Computing in Applied Probability*, Vol 1(No. 1), 1999.

[6] Qiu D. and Srikant R.J. Modeling and performance analysis of bittorrent-like peer-to-peer networks. Technical report, University of Illinois at Urbana-Champaign, USA, 2004.

[7] Tsoumakos D. and Roussapoulos N. A comparison of peer-to-peer search methods. *International Workshop on the Web and Databases (WebDB), San Diego, California, USA*,, 2003.

[8] Ralph B. D'Agostino and Michael A. Stephens, editors. *Goodness-of-fit Techniques*. Dekker, 1986.

[9] David Erman, Dragos Ilie, and Adrian Popescu. Peer-to-peer traffic measurements. Technical report, Blekinge Institute of Technology, Karlskrona, Sweden, 2005.

[10] David Erman, Dragos Ilie, Adrian Popescu, and Arne A. Nilsson. Measurement and analysis of BitTorrent traffic. In *NTS 17*, August 2004.

[11] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[12] Dragos Ilie, David Erman, and Adrian Popescu. Traffic measurements of P2P systems. *Swedish National on Computer Networking Workshop (SNCNW04)*, November 2004.

[13] Dragos Ilie, David Erman, Adrian Popescu, and Arne A. Nilsson. Measurement and analysis of Gnutella signaling traffic. In *IPSI 2004*, September 2004.

[14] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *PAM2004*, 2004.

[15] Pouwelse J.A., Garbacki P., Epema D.H.J., and Sips H.J. The BitTorrent P2P file-sharing system: Measurements and analysis. *4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.

[16] Van Jacobsen, Leres C., and McCanne S. Tcpdump. http://www.tcpdump.org.

[17] Ajit K. Jena, Adrian Popescu, and Arne A. Nilsson. Modeling and evaluation of internet applications. In *International Teletraffic Conference*, Berlin, Germany, August 2003. ITC18.

[18] Kurose J.F. and Ross K.W. *Computer Networking, A Top-Down Approach Featuring the Internet*. Addison-Wesley, 2003. ISBN 0-201-97699-4.

[19] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Claffy K, and Michalis Faloustos. File sharing in the internet: A characterization of P2P traffic in the backbone. Technical report, University of California, Riverside, USA, 2003.

[20] Tor Klingberg and Raphael Manfredi. *Gnutella 0.6*. The Gnutella Developer Forum (GDF), 200206-draft edition, June 2002. http://groups.yahoo.com/group/the_gdf/files/Development/.

[21] Balachander Krishnamurty and Jennifer Rexford. *Web Protocols and Practice*. Addison Wesley, 2001. ISBN 0-201-71088-9.

[22] Peterson L.L. and Davie B.S. *Computer Networks: A System Approach*. Morgan Kaufman, 2003. ISSB 1-55860-833-8.

[23] Napster. Napster. http://www.napster.com.

[24] Sharman Networks. KaZaA. www.kazaa.com, February 2005.

[25] Shawn Ostermann. Tcptrace. http://www.tcptrace.org.

[26] SETI@Home – the search for extraterrestial intelligence. http://setiathome.ssl.berkeley.edu/, February 2005.

[27] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, 1985. ISBN 0-471-90763-4.