# BitTorrent Request Message Models

David Erman
Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
371 79 Karlskrona, Sweden
david.erman@bth.se

Adrian Popescu
Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
371 79 Karlskrona, Sweden
adrian.popescu@bth.se

***Abstract*— BitTorrent, a replicating Peer-to-Peer (P2P) file sharing system, 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. This paper reports on new results obtained on modelling and analysis of BitTorrent traffic collected at Blekinge Institute of Technology (BTH) as well as a local Internet Service Provider (ISP). In particular, we report on new request models for a BitTorrent peer during downloading.**

## I. Introduction

The Internet has experienced two major revolutions. The first was the emergence of the World Wide Web, which catapulted the Internet from being a scientific and academic network to becoming part of the societal infrastructure. The second revolution was the appearance of the Peer-to-Peer (P2P) applications, spear-headed by Napster.

The popularity of P2P networking has lead to a dramatic increase of the volume and complexity of the traffic generated by P2P applications. P2P traffic has recently been shown to amount to over 92 % of the total traffic in transit and peering links [1]. One of the major contributors to this massive volume of traffic is BitTorrent, a P2P replication system. Studies have shown that BitTorrent traffic increased from 26 % to 52 % of the total P2P traffic volume during the first half of 2004 [2], and still amounts to 60 % in 2005 [1]. Despite this, measurement studies and analysis of BitTorrent traffic have been rather limited so far. This is likely because of the complexity of this task, which involves answering hard questions related to data retrieval and content location, storage, data analysis and modelling of traffic and topological characteristics as well as privacy and copyright issues.

The primary goal for this paper is to understand the request characteristics of a BitTorrent peer, and to present models suitable for use in a simulation environment. Together with previously reported models of session characteristics and other characteristics such as response times and download times [3], we aim at having a comprehensive set of models for the BitTorrent system. Furthermore, we also hope that our models will help in better forecasting the traffic increase in the Internet infrastructure given the immense popularity of BitTorrent.

In general, measurement studies of P2P systems are limited in number. For instance, Saroiu *et al.* performed a measurement study of Napster and Gnutella in 2002 [4]. Active and passive measurements were performed on both systems. Their results show the non-cooperativity of peers involved in the systems and several other characteristics such as estimated peer bandwidths, number of shared files and resilience.

The present work is one of a few works investigating the properties of the BitTorrent system. For instance, in [5], the authors use tracker and client logs to evaluate the performance on both global and session scales. They note the efficiency of the *tit-for-tat*-policy employed in BitTorrent, and the flexibility and scalability of the protocol.

Qiu and Srikant present a fluid flow model for BitTorrent-like file-sharing P2P networks [6]. They assume a Poisson peer arrival process and exponentially distributed download times. Additionally, the authors assume that seeds remain in the network according to an exponentially distributed time and identical download rates for all peers. Their results state that the number of seeds and leechers are Gaussian random variables when in steady state.

In this paper, we report on new models for the request behaviour of a downloading BitTorrent peer, obtained at BTH and a local ISP. As all BitTorrent protocol messages are fixed-size, both the request message process and the corresponding response message process constitute implicit bitrate models.

The rest of this paper is as follows: Section II briefly describes the BitTorrent system. In Section III, we present an overview of the measurements used for the models presented in this paper. Section IV continues by discussing the modelling methodology employed, followed by a presentation of request models in Section V. Section VI then concludes the paper.

## II. BitTorrent

BitTorrent is a P2P protocol for content distribution and replication designed to quickly, efficiently and fairly replicate data [7], [8]. It uses swarming to distribute load among participating peers. A BitTorrent network, also denoted as a BitTorrent *swarm*, does not provide any resource query, lookup, routing or topology forming functionality. The sole purpose of a BitTorrent swarm is to efficiently disseminate data.

A swarm consists of *peers* and at least one *tracker.* The role of the tracker is to provide peers with IP addresses to other peers; it does not participate in the content distribution. Peers are partitioned into *seeds* and *leechers*. A seed is a peer that is in possession of the entire content of the swarm, while a leecher is a peer that still needs to download parts of the content. An initial seed is necessary for a swarm to be successful.

The leech phase of a BitTorrent peer may be partitioned into three different sub-phases. During the first sub-phase, the peer is trying to connect to a predefined maximum number of other peers. This means that the number of connected peers increases during this phase, thus also increasing the number of outgoing piece requests. On entering the second phase, the peer has connected to enough peers. The number of connected peers and outgoing messages fluctuate around some average value during this phase. The final phase is the end-game mode, during which the peer sends a large amount of requests.

The content distributed in a BitTorrent swarm is divided into *pieces*. Data is requested among peers using piece numbers and byte ranges in these pieces as identifiers. The byte ranges are also known as *subpieces* or *chunks*.

To join a specific BitTorrent swarm, a potential peer must first acquire a set of metadata, known as a *torrent file.* The torrent file contains, among other information, the address to the swarm tracker and a set of Secure Hash Algorithm One (SHA1) hash values for

the content pieces. The SHA1 hash for the torrent file itself acts as an identifier of the swarm and is used in both peer handshakes and tracker queries.

Fairness in the BitTorrent system is implemented using a scheme in which peers express desire to download by sending *interested*-messages to peers that have pieces needed. The serving peers may then allow or disallow download with *unchoke*- or *choke*-messages respectively. Data transfer then takes place by the downloading peer issuing *request*-messages to randomly selected serving peers and serving peers responding with *piece*-messages.

Figure 1 depicts a typical protocol exchange for a BitTorrent download during which peer A downloads from peers B and C.

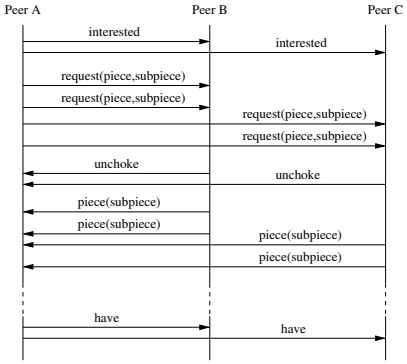



Fig. 1. BitTorrent download procedure

### A. End-game Mode

End-game mode occurs when a peer only has a few pieces of the content left to download. During early development of BitTorrent, downloads showed a tendency to stall around 99 % completion [9]. To solve this, the peers were programmed to request the remaining pieces from *all* connected serving peers, instead of random subsets of them. The pieces are then downloaded from the peers that respond quickest, and *cancel*-messages are sent to the remaining peers.

## III. Measurements

The measurements performed for the models presented in this paper have previously been reported in [3], [10].

The P2P research group at BTH has implemented a dedicated measurement infrastructure to measure application layer messages at the link layer. This infrastructure contains both TCP reassembly and application message stream decoding, and was used for collecting the traffic traces reported in this paper. Details on the BTH measurement infrastructure are reported in [11].

A number of 13 measurements have been performed in 3 measurement sets at 2 locations. Measurements 1–8 (measurement set 1) and 13 (measurement set 3) were performed at BTH, while the remaining measurements were performed at a local ISP (measurement set 2). Measurements 1–12 were collected as application logs from an instrumented version of the reference BitTorrent client. Measurement 13 data was collected using both link layer packet capture and application logs. Due to hardware failure, the data for measurement 9 was irretrievably lost.

Measurement sets 1 and 2 were performed in May 2004, with each measurement ranging in duration from just under 3 days to 1 week, with some temporal overlap (Figure 2). Set 3 was performed during 1 week in June 2004. In total, about 1.4 GB of compressed log files and 143 GB of packet traces were collected.

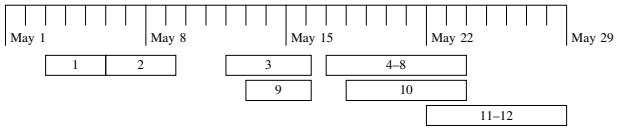



Fig. 2. Temporal structure of measurements 1–12

To avoid copyright issues, the data content of the measured swarms was several Linux distributions, ranging in size from 650 MB to 4.3 GB. (Table I)

TABLE I
Content summary

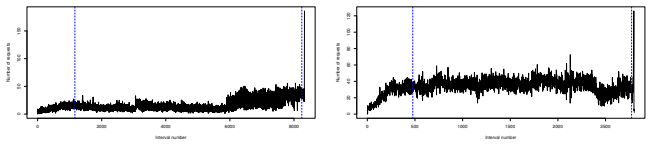
| Content | Size | Measurement |
|---|---|---|
| RedHat FC 2 test3 CD Images | 2.2 GB | 1–3 |
| RedHat FC 2 test3 DVD Image | 4.3 GB | 6, 10 |
| Slackware Linux Install Disk 1 | 650 MB | 4 |
| Slackware Linux Install Disk 2 | 670 MB | 5 |
| Dynebolic Linux 1.3 | 650 MB | 7, 9 |
| Knoppix Linux 3.4 | 700 MB | 8 |
| RedHat FC 2 ‘Tettnang‘ CD Images | 2.2 GB | 12,13 |
| RedHat FC 2 ‘Tettnang‘ DVD Image | 4.3 GB | 11 |

## IV. Modelling methodology

### A. Censoring

A study on measurement and modelling of BitTorrent messages for the link layer traces of measurement 13 is reported in [3]. For the upstream leech phase request rate models reported there, a subset of the data was used for modeling to ensure that the end-game mode and startup-phase of the peer do not skew the model. For these models, the censoring was done to identify the steady-state behaviour of the request process.

For the models presented in this paper, a heuristic procedure is used to locate the steady-state portion of the request rate time series. The upper limit is set at 99 % of the time series, thus excluding the end-game mode requests. To locate the lower limit, we calculate a smoothed version of the time series using a Lowess smoother [12]. The lower limit is then defined as the location of the first minimum of the numerical derivative of the smoothed time series. Figure 3 shows examples of two typical request rate time series, with the upper and lower limits indicated by dashed lines.



(a) Measurement 12 (b) Measurement 13

Fig. 3. Upstream request rates

### B. Model selection

As BitTorrent request rates have been shown to be well modelled by a Gaussian distribution [3], we have opted to employ a heuristic procedure using several “Gaussian-like” distributions to identify the models for this work. The distributions involved are the Gaussian, Weibull, Log-normal, a binary mixture distribution of Weibull and

Log-normal as well as *dual* versions of the Gaussian and Weibull distributions. The general form for a binary mixture distribution is

$$f(x;\Theta_1,\Theta_2) = pf_1(x;\Theta_1) + (1-p)f_2(x;\Theta_2)$$

where $f_1$ and $f_2$ are the *component densities* with associated parameters $\Theta_1$ and $\Theta_2$, and $p$ is the *mixing weight*. By a dual distribution we refer to a binary mixture distribution, in which the component distributions are the same, i.e., $f_1(x) = f_2(x)$.

In this paper, we use the model notation shown in Table II, in which $\mu$ and $\sigma$ are the means and standard deviations for the associated distribution or component distribution, and $\alpha$ and $\beta$ are the shape and scale parameters.

TABLE II
MODEL NOTATION

| | |
|---|---|
| Gaussian | $G(\mu,\sigma)$ |
| Log-normal | $LN(\mu,\sigma)$ |
| Dual Gaussian | $DG(\mu_1,\sigma_1,\mu_2,\sigma_2,p)$ |
| Dual Weibull | $DW(\alpha_1,\beta_1,\alpha_2,\beta_2,p)$ |
| Log-normal – Weibull | $LW(\mu,\sigma,\alpha,\beta,p)$ |

The heuristic procedure uses both the Maximum Likelihood Estimation (MLE) and minimum distance methods for parameter estimations [3]. For each measurement, the following procedure is performed

1) Employ MLE to find initial parameter estimates for the Gaussian and Weibull distribution.
2) Further minimise the estimation error using the minimum distance method (i.e., minimise over the error percentage described in Section IV-C).
3) Select the distribution and associated parameters, which resulted in the lowest $E_{\%}$.
4) If the minimum $E_{\%}$ is larger than 2, inspect the associated measurement data manually. In certain cases, the optimisation algorithm used fails to locate the global minimum. In this case the starting values for the optimisation are modified manually.

### C. Fitness Assessment

Classical hypothesis tests such as the Kolmogorov-Smirnov (KS) and Anderson-Darling (AD) tests tend to reject the null hypothesis, i.e., reject a selected distribution with associated parameters, when the number of observations is large [13]–[15]. The reasons for this is because of model inconsistencies, estimation errors or the presence of long-range dependence in the data.

While the test statistics used in hypothesis tests are still useful as fitness *measures*, i.e., for use in relative fitness assessment, they are less suitable for use in deciding whether or not to reject a particular estimated distribution. To determine the fitness of the models reported in this paper, we employ an error percentage absolute fitness measure. The measure, denoted by $E_{\%}$, uses the Probability Integral Transform (PIT) method to transform the test for a specific distribution to a test for uniformity. We then use the fitness classes in Table III to accept or discard a specific estimated distribution.

TABLE III
FITNESS QUALITY BOUNDARIES

| $E_{\%} \approx$ | 0 | 1 | 2 | 3 | 4 | $\geq 5$ |
|---|---|---|---|---|---|---|
| Degree | excellent | very good | good | fair | poor | fail |

The expression for calculting the error percentage is

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

where $E_{max}$ is defined as

$$\int_0^1 \sup\{U(x), 1-U(x)\}\ \mathrm{d}x = \frac{3}{4} \tag{2}$$

and the $\hat{U}_i$ are the observations from the PIT transformed estimated distribution.

It is important to mention that this is *not* a statistical significance level, but rather an acceptable margin of error. The error percentage is further discussed in [3].

## V. REQUEST MODELS

This section reports on models for upstream request rates during the downloading, or *leech*, phase of a BitTorrent peer.

The resolution for the models reported in this section is 1 s. This resolution has been chosen partly to reduce the amount of data in larger sample sets and partly due to the difficulties associated with computation of instantaneous rates on short timescales.

The *request*-messages and their responses are the major bandwidth contributors in a BitTorrent session. Modelling the request behaviour of the measurement peer provides valuable information to describe the overall behaviour of the entire swarm.

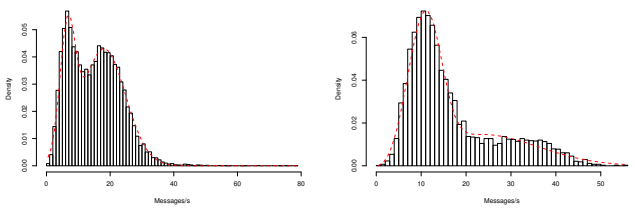
TABLE IV
REQUEST RATE MODELLING RESULTS

| Measurement | Model and parameter estimates | Error | Comment |
|---|---|---|---|
| 1 | $DG(88.7, 13.0, 88.3, 8.55, 0.11)$ | 0.76 % | Pass, VG |
| 2 | $DG(78.0, 15.5, 78.4, 6.47, 0.61)$ | 0.64 % | Pass, VG |
| 3 | $LW(7.5, 114.8, 4.07, 6.45, 0.90)$ | 1.18 % | Pass, VG |
| 4 | $DW(4.32, 20.9, 6.01, 17.0, 0.51)$ | 0.55 % | Pass, VG |
| 5 | $DG(15.2, 4.63, 13.1, 2.33, 0.55)$ | 0.49 % | Pass, E |
| 6 | $LN(2.63, 0.25)$ | 0.79 % | Pass, VG |
| 7 | $DG(40.2, 13.6, 41.0, 12.3, 0.22)$ | 1.13 % | Pass, VG |
| 8 | $DW(3.10, 9.90, 4.26, 7.76, 0.60)$ | 0.53 % | Pass, VG |
| 10 | $DG(15.0, 5.39, 9.27, 3.11, 0.40)$ | 0.91 % | Pass, VG |
| 11 | $DG(18.0, 6.79, 6.62, 2.38, 0.73)$ | 0.30 % | Pass, E |
| 12 | $DW(2.5, 28.9, 3.49, 12.0, 0.42)$ | 0.72 % | Pass, VG |
| 13 | $DW(5.32, 55.8, 10.7, 54.8, 0.51)$ | 0.93 % | Pass, VG |
| 13 [a] | $DW(7.60, 36.6, 5.67, 39.8, 0.29)$ | 0.39 % | Pass, E |

[a] Modelled from Ethernet traces

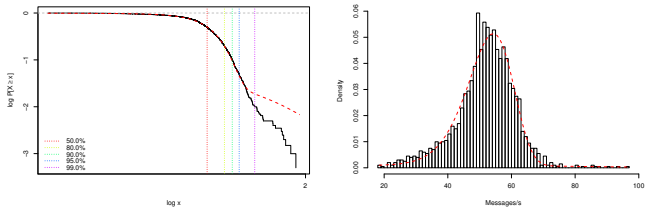
Table IV presents the results of the modelling procedure. An important observation is that dual models clearly dominate. In certain cases, such as for measurements 11 and 12 (Figure 4), the bi-modality of the data indicates the applicability of a binary model. The modes correspond to a changing number of outgoing requests due to an increase or decrease in number of connected peers. This change in number of requests is shown in Figure 3(a).

For some of the remaining dual models, we believe that the reason for the appearance of a mixture distribution is due to a skew in the distribution body mass, such as shown in Figures 5(b) and 5(d). It is interesting to note the appearance of a single Log-normal distribution for measurement 6. The content for this measurement was the same as for measurement 10, but the measurement was performed at a different location. Measurement 10 was performed for a slightly

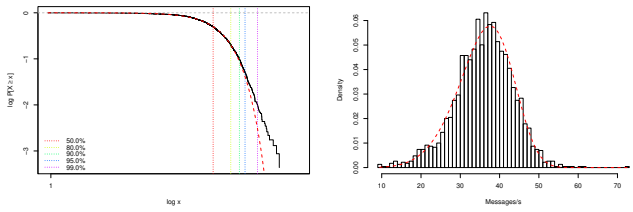



(a) Measurement 11 (b) Measurement 12

Fig. 4. Bi-modal distributions




(a) Application log CCDF (b) Application log EPDF




(c) Link layer trace CCDF (d) Link layer trace EPDF

Fig. 5. Measurement 13 modelling results

shorter period of time, which may affect the results. Measurement 6 was started about 1 day before measurement 10, and during this time the number of connected peers decreased significantly.

An important issue to consider when performing measurements on application layer messages is the influence of the operating system and TCP/IP stack on the measured timestamps. The timings of the protocol messages are affected by buffering in both the TCP/IP stack and the runtime environment of the application. Both TCP/IP stack and runtime environment add buffering and thus delays into the application message process. Additionally, system load may affect the responsiveness of the application and the TCP/IP stack. Measurement 13 was performed by collecting traces from both application logs and captured packets, which provides the opportunity of assessing the influence of TCP/IP stack and application runtime. For instance, Figure 5 shows the EPDF and CCDF for both link layer packet captures and application logs for measurement 13. The fact that similar models apply to both cases is encouraging, and indicates that, for the considered resolution, the impact of the operating system and TCP/IP stack is negligible. However, this does not imply that the impact of the TCP/IP stack can be ignored on shorter timescales, as a one second resolution is large compared to the timescales at which a CPU works. This makes e.g., network performance modelling for inter-departure times difficult to perform by using only application logs, since the true network performance measures are the result of messages traversing both the application runtime environment and TCP/IP stack. In effect, the TCP/IP stack and application runtime act as a filter to the message departure process. Fortunately, packet traces do not suffer as much from the same problem, as the application messages are measured using link layer timestamps after exiting the stack, thus measuring the network performance *after* the filtering has taken place. The impact of the TCP/IP stack is even more pronounced in high-speed networks, where it becomes a major bottleneck.

## VI. Conclusions

This paper has reported on new models for request rates during the downloading phase of a BitTorrent peer.

The results show that, in most cases, the request rate distributions are composed of a mixture of two distributions, primarily dual Weibull and Gaussian distributions. We believe that this is because of a strong correlation existent between the number of distributions or modes and the number of connected peers. This is however subject to further research, as is the extent of these correlations.

Planned future work is to perform similar analysis for *incoming* request rates during the seeding phase of the measurement peer. Also, BitTorrent peers tend to send several requests back-to-back to allow TCP to increase throughput. Analysis of these batch request arrivals would provide valuable insight into the network performance of a BitTorrent peer. It is our ambition to ultimately report on a rather complete set of BitTorrent models.

## References

[1] C. A. Parker, "P2P in 2005." http://www.cachelogic.com/research/slide9.php, May 2005.
[2] C. A. Parker, "The true picture of peer-to-peer file sharing." http://www.cachelogic.com/research/slide9.php, May 2005.
[3] D. Erman, "Bittorrent traffic measurements and models," October 2005. Licentiate thesis, Blekinge Institute of Technology.
[4] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of the Multimedia Computing and Networking (MMCN)*, January 2002.
[5] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five months in a torrent's lifetime," in *PAM2004*, 2004.
[6] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," tech. rep., University of Illinois at Urbana-Champaign, USA, 2004.
[7] B. Cohen, "BitTorrent." http://bitconjurer.org/BitTorrent/, August 2005.
[8] "BitTorrent specification." http://wiki.theory.org/BitTorrentSpecification, February 2005.
[9] B. Cohen, "BitTorrent protocol specification." http://www.bitconjurer.org/BitTorrent/protocol.html, February 2005.
[10] D. Erman, D. Ilie, and A. Popescu, "Peer-to-peer traffic measurements," tech. rep., Blekinge Institute of Technology, Karlskrona, Sweden, 2005.
[11] D. Ilie, D. Erman, and A. Popescu, "Traffic measurements of P2P systems," *Swedish National on Computer Networking Workshop (SNCNW04)*, November 2004.
[12] W. Cleveland, "Robust locally weighted regression and smoothing scatter plots," *Journal of American Statistical Association*, vol. 74, pp. 829–836, 1979.
[13] R. B. D'Agostino and M. A. Stephens, eds., *Goodness-of-fit Techniques*. Dekker, 1986.
[14] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill, 2000. ISBN 0-07-059292-6.
[15] J. Beran, *Statistics for Long-Memory Processes*. Chapman & Hall, 1994.