Performance Analysis of BitTorrent Protocol

Parul Sharma
Department of Computer Science Engineering
Indira Gandhi Institute of Technology
GGSIPU, Delhi, India.
sharma.parul111@gmail.com

Anuja Bhakuni
Department of Computer Science Engineering
Indira Gandhi Institute of Technology
GGSIPU, Delhi, India.
anuja.bhakuni13@gmail.com

Rishabh Kaushal
Department of Information Technology
Indira Gandhi Institute of Technology
GGSIPU, Delhi, India.
rishabh.kaushal@gmail.com

Abstract— **BitTorrent protocol has been quite successful in capturing majority of Internet traffic since its inception due to its various mechanisms to improve the download performance of the network. The mechanisms like tit-for-tat, optimistic unchoking, anti-snubbing and various piece selection strategy, etc. have been working very well. This work is an attempt to study the performance of BitTorrent protocol and its networking infrastructure through two strategies, namely, (A) Simulations based using BitTorrent patch in ns-2 simulator and (B) Real networks using client implementation based on *libtorrent* library. Empirical results thus obtained are used to infer various performance parameters of BitTorrent protocols.**

Keywords — BitTorrent protocol, Peer to Peer (P2P), Internet, Networks.

I. INTRODUCTION

Ever since the development and subsequent release of BitTorrent protocol by its inventor Bram Cohen in July, 2001 [1], it has remained a very popular file sharing protocol. In November 2004, BitTorrent accounted for a huge 35 percent of all the traffic on the Internet and in 2006 the BitTorrent protocol has risen to over 60 percent of all Internet traffic according to British Web analysis firm CacheLogic and 78 % of all the global p2p traffic [2]. As per a recent article in Jan., 2012, it is claimed that BitTorrent is accounting for an unbelievable 150 million monthly users [3]. Furthermore, it is estimated that BitTorrent, at any given point of time, has on an average more active users than YouTube and Facebook combined [4] [5].

One key aspect of performance in BitTorrent is the download rate that is achieved by participating peers. In this paper we try to find the relation of download speed with various parameters like upload bandwidth, swarm size , start-time, first-chunk-time etc. It is done by performing p2p simulations through NS2 simulator and also through libtorrent-rasterbar in the real environment.

The remainder of paper is organized as follows. Working of the BitTorrent protocol is explained in Section II. Various mechanisms like peer and piece selection strategies and the historic work done are presented in section III and section IV respectively. In Section V, we present our empirical analysis of BitTorrent network. Finally conclusion is drawn in section VI.

II. WORKING OF BITTORRENT PROTOCOL

BitTorrent protocol was primarily designed aiming to be a substitute of old centralized HTTP downloads, and not a full p2p protocol. But as it improved, it became peer-to-peer in nature where users connected to each other directly to upload and download portions of a large file (called as a piece) from other peers who have also downloaded either the file or parts of it. These pieces are, henceforth, reassembled into the full file. This mechanism provided faster download whereas in normal p2p systems, the peer downloads from a single peer alone and the download speed is limited. The conventional centralized server based form of downloads were marred with problems of server bottlenecks and also contributed to network congestion. While BitTorrent was designed to provide a faster download and congestion free mechanism. The property of simultaneous uploading while downloading ensures that there is no single source to download from (thereby removing the bottleneck problem) and also ensures that network traffic due to file download gets distributed across the network (thereby preventing the chances of network congestion)[6].

To start a BitTorrent deployment, a *.torrent metadata file* is put on an ordinary *web server/portal*. The *.torrent* consists of information about the file, its name, length, URL of a tracker and hashing information. Tracker helps downloader(s) find each other. A very basic protocol called HTTP tracker protocol is used in which a downloader sends a message including information about downloading file, the port used etc. and the tracker replies with a list of peers (including contact information) downloading the same file. Downloader(s) then use this information to connect to each other.

To make a file available, a *seed*, i.e., a 'downloader' which already has complete file starts first . The bandwidth

978-1-4673-5952-8/13/$31.00 © 2013 IEEE

requirements of the *tracker* (which is a server that maintains a list of clients forming the BitTorrent swarm associated with given content and is also aware of the download progress of each peer within the swarm) and *web server* are very low, while the *seed* must send out at least one complete copy of the original file. For downloading the files that are hosted using BitTorrent users must have a *BitTorrent client/software* (a computer program designed for p2p file sharing using the BitTorrent protocol). The *leechers,* not having all pieces of a large file, have to upload the pieces it has, and download the ones requested from either *leechers* or *seeders* to reassemble these into a complete file. All these users (*seeders & leechers*) together combine to form a *swarm.* The download bandwidth load is divided among many sources since the users download from several other users and not from a single central server. This benefits not only the downloader by increasing its download rates but also effectively reduces the cost of upload bandwidth for servers hosting large files, since BitTorrent protocol effectively distributes the upstream bandwidth of a single server across all downloaders, hereby gaining advantage from peers for the whole BitTorrent system.

III. MECHANISMS USED IN BITTORRENT PROTOCOL

The mechanisms employed by BitTorrent mainly consist of peer and piece selection strategies [7][8]. A superlative peer selection strategy must maximize the service capacity of the system and a good piece selection strategy must ensure that each peer can find pieces that it requires from its neighbors. They play a major role in determining the downloading experience of any peer.

A. Peer Selection Strategy:

The peer selection strategy of BitTorrent uses four principal mechanisms: tit-for-tat, optimistic unchoking , upload only and anti-snubbing. Collectively, they form the base for the choking algorithm utilized by a peer. The main aim of these mechanisms is to improve the downloading performance of those peers that help induce the file exchange by uploading pieces to other peers, and punish freeriders.

i. Tit-for-tat:
The tit-for-tat strategy is taken up by a peer to preferentially upload to its neighbor peers who provide it the optimum downloading rates. At any given time, a peer exchanges file pieces with a fixed number of neighboring peers. A leecher preferentially uploads to the best neighbors who provide it the best downloading rate and chokes others. Every few seconds, the leecher reevaluates the downloading rate from all peers who are sending data to it. If another peer offers better downloading rate, the leecher will choke the peer with the smallest downloading rate among the current peers it is serving, and unchoke the better peer. This mechanism is very important to encourage downloaders and punish free-riders, thus preventing leechers from downloading without uploading anything.

ii. Optimistic Unchoking:
BitTorrent incorporates an optimistic unchoking policy, besides the TFT mechanism. Every few seconds, a peer unchokes a neighboring peer randomly regardless of its uploading rate. The benefit of Optimistic Unchoking is that it enables a peer to discover better neighboring peers to exchange file pieces with since other neighbors may have higher capacity for uploading than the currently unchoked peers. If a peer uses only the TFT mechanism, there will be no opportunity for discovering other peers that can provide higher uploading rate. This strategy is extremely useful for newly joined peers to get started.

iii. Upload Only:
A peer becomes a seed, once it finishes downloading the entire file. Seeds cannot select peers based on downloading rates, as seeds have nothing to download. The seeds prefer to upload to peers with better uploading rates.

iv. Anti-snubbing:
A peer may be choked by the peers it was earlier downloading from, thereby getting poor downloading rates. To address this problem, when a peer notices that some time has elapsed without getting a single piece from a neighboring peer, the leecher assumes it is 'snubbed' by that peer and does not upload to it any further through regular unchoke.

B. Piece Selection Strategy:

The second main strategy that plays a significant role in the success of BitTorrent is the piece selection strategy. When a peer wants to download pieces from its neighbours, it adopts several piece selection strategies, those mechanisms are as follows:

i. Strict Priority:
In BitTorrent, peers concentrate on downloading a whole piece of a file, before requesting for another piece.Thus if a sub-piece is requested, then subsequent subpieces of the same piece will be requested preferentially so as to complete the download of the whole piece as soon as possible, as only complete pieces can be exchanged with others.

ii. Endgame Mode:
This mode is taken up by a peer in the end of downloading the file. If a piece is requested from a peer that has a slow transfer rate, the downloading time gets prolonged. To address this problem, a peer requests all of its neighbours for blocks it has not received. After a block is obtained, the peer aborts the request for that block from its neighbours, so as to lessen the bandwidth wastage due to redundant downloads.

iii. Rarest First:
Peers generally prefer to download pieces that are the rarest among their neighbours. This strategy is known as the rarest first algorithm which works as follows. Every peer maintains a list of the pieces that each of its neighbours contains. This list is updated every time a copy of a piece becomes available from its neighbours. It then creates a rarest-pieces set which is the list of those pieces that have the minimum number of copies among its neighbours. It then chooses the piece to download that is rarest among its neighbours. This strategy increases the chance of a peer exchanging with its neighbours as it has pieces that others require. This also benefits the seeds by reducing their server burden, since they need to upload less copies of the same piece. Furthermore, the exit of a seed will not leave the remaining

downloaders without the opportunity to exchange file pieces, because this strategy assures that all the pieces are quickly distributed to maximum of the leechers.

iv. Random First Piece:

An exception exists to the rarest first policy when a peer first joins a torrent. Since the peer does not have any pieces, so it would like to download a whole piece quickly so that it can get ready to reciprocate with the TFT algorithm. In this case, the new peer will not take the rarest piece. Instead, it will download the first piece randomly in order to make sure it can get a whole piece as soon as possible.

IV. RELATED WORK

A lot of studies have been performed on the measurements and modeling BT performance affected by various parameters. Izal et al. [9] presented the first comprehensive evaluation of BitTorrent using measurement and used the *tracker* log of Linux RedHat 9 distribution for their evaluation. Here, they indicated that the uploading and downloading rates are correlated, which verified the effectiveness of *TFT policy*. Differing from Izal [10] testing only on a single *torrent*, Legout et al. [11] conducted a detailed measurement study aimed at understanding the behavior of the *rarest-first algorithm* and the *choking algorithm* on several real torrents. Chih-Lin HU et al. [12] investigated on a peer-side basis in terms of downloading side bandwidth allocation and usage with various parameters like bandwidth allocation and file size for real traffic traces. Bharambe et al. [13] focused on two main metrics: utilization of upload capacity and (un)fairness in terms of data size served by node and hence evaluated the impact of BitTorrent's mechanisms on swarm performance under a variety of flash-crowd workloads using a simulator, which models both peer's activity and BitTorrent's mechanisms like *TFT* and *rarest-first*. Felber et al. [14] presented the effect of different peer and piece selection strategies on the performance of BitTorrent using simulations Urvoy-Keller et al. [15] shows the significant impact of two overlay topology parameters, *peer set size* (PS) and percentage of *outgoing connections*(OC) on BitTorrent's performance using a simulation approach.

Differing from the studies above where the focus is on upload efficiency and overall system performance, we analyze the download performance of a single user (rather than a whole swarm) against parameters like upload rate, number of peers and first chunk time using *ns2 simulator* and real-time *libtorrent* client to verify various piece and peer selection strategies affecting performance of the downloader.

V. EMPIRICAL ANALYSIS OF BITTORRENT NETWORK

In this section, we present an extensive empirical analysis of the BitTorrent protocol and its supporting networking infrastructure. Two strategies have been adopted for the empirical analysis, namely, (A) Simulation based analysis, which was done using *ns-2 simulator* [15][16] and (B) Real-network analysis which was done using client example provided by the open-source *libtorrent* library compiled with qTorrent option as GUI.

A. Simulations using ns2

We used *ns2 simulator* [15] to test the BitTorrent protocol and the effects that the chosen parameters have on the download time performance of swarm. To facilitate our analysis, we have used bt_flashcrowd_star.tcl (provided by [16]) file to create a BT network which essentially checks the efficiency of data transfers between peers, ignoring checking the download data integrity by hash values. We chose the ns2 based BitTorrent implementation since it supports various input-output parameters and is also easy to understand and simple to use. As per this implementation, unchoking algorithm is implemented while anti-snubbing and endgame features of BT protocol were ignored. Super-seeding which improves flash crowd scenario has been implemented. The peer and piece selection algorithm can be replaced by alternatives being modular. The ns-2 simulator was patched with BitTorrent (BT) patch which creates 4 additional classes to support basic BitTorrent protocol operation, namely, *BitTorrent Application*, *BitTorrent Tracker*, *BitTorrent Connection* and *BitTorrent message*. BT patch uses full duplex TCP (bidirectional) as the carrier transport layer protocol for simulating BitTorrent traffic using *ns-2 simulator*.

Following three input parameters which were tested to verify the outcome:

i. Number of peers.
ii. Seed with random number.
iii. Download Rate of the network (in Kbps).

The output log file generated by *ns-2 simulator* contained the following output parameters which were verified against input parameters:

i. Peer ID, unique ID of the peer.
ii. Start Time, time when the peer joins the BitTorrent network.
iii. First Chunk Time, time when the peer receives its first piece of file.
iv. Download Finish Time, time when the peer completes its download.
v. End Time, time when the entire swarm finishes downloads.
vi. Total Download Time, is calculated as the difference between Download Finish Time and Start Time.

Keeping number of peers as constant (configured to a value of 10), analysis was done to ascertain when the peer gets its first chunk with respect to the time when it started its download. Download rate was varied during the run with different values - 500 Kbps, 1000 Kbps, and 2000 Kbps. As evident in Figure 1, it is observed that the first chunk time of the fastest network rate (i.e. = 2000 Kbps) was the least, that is the peers got their first piece in the fastest time. But varying this rate to 1000 Kbps didn't have much effect on the first chunk time. But when the rate dropped down to 500 Kbps, the first chunk time increased considerably. Conclusion: *Keeping download rate fixed, the time when the peer receives its first piece of file (chunk) is proportional to when it joins the network (start time) i.e. the first chunk time increases with rise in start time during the initial run of network when the load is less, however, afterwards, this trend no longer exist when the network load increases.*

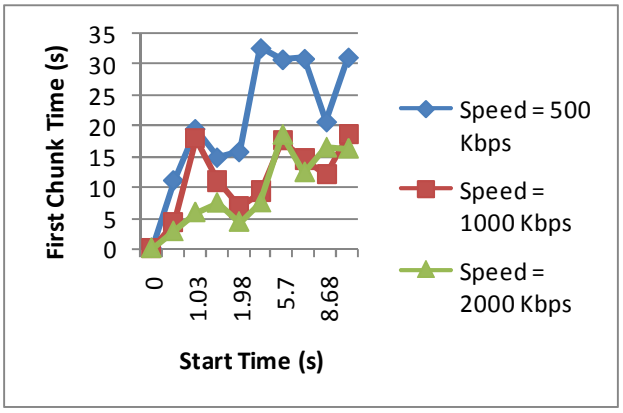

Figure 1: Start Time vs. First Chunk Time (# peers = 10)

In the next analysis, download finish time was compared with the start time for each peer (again keeping number of peers fixed to 10) with varying download rate of 500 Kbps, 1000 Kbps and 2000 Kbps, as show in Figure 2. Conclusion: *It is observed that that the download finish time for each peer remains constant, in other words, it can be concluded that it is independent of download start time (keeping download rate constant).* It is due to the fact that peers joining late in the network have more file pieces distributed across multiple peers available for parallel downloads. However, by increasing the download rate from 500 Kbps to 2000 Kbps, the Download Finish Time of a single peer decreases. Conclusion: *Download Finish Time improves (reduces) by increasing download rate.*

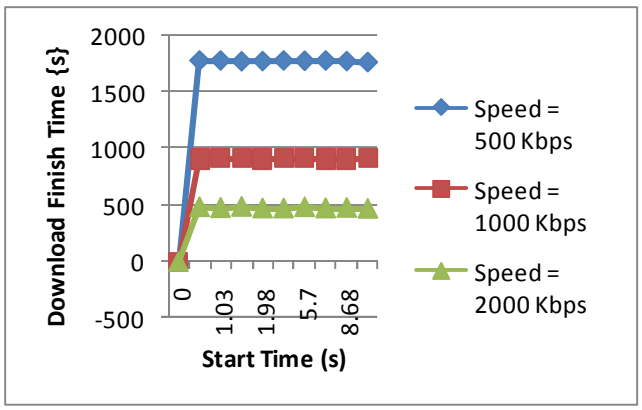

Figure 2: Start Time vs. Download Finish Time (#peers = 10)

Next, Total Download Time was studied with respect to the Start Time keeping different values of download rates. In this case, observations are in consonance with the earlier ones. Conclusion: *The Total Download Time of each peer remains same as long as download speed is not varied. Furthermore, even Total Download Time is invariant to the Start Time and increases with an increase in speed of the network.*

The Download Finish Time (Figure 3) and Total Download Time were verified against the First Chunk Time, with varying download rate of 500 Kbps, 1000 Kbps, and 2000 Kbps. Similar to the Start Time analysis, *it was observed that First Chunk Time doesn't have significant effect on the Download Finish Time and the Total Download Time*, which more or less remained constant within the same download rate level (say 484 when speed = 500 Kbps) and increases approx. 100 % every time speed is doubled.

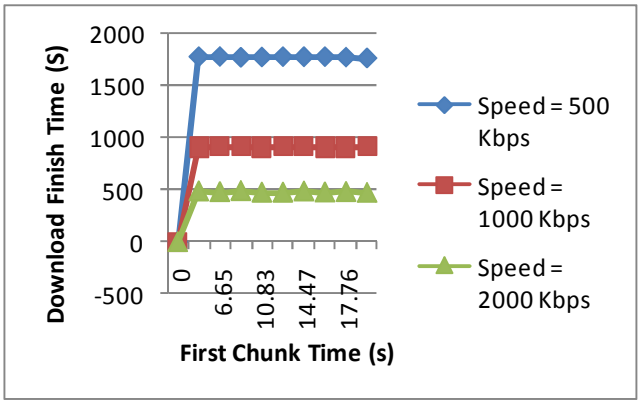

Figure 3: First Chunk Time vs. Total Download Time

The Download Finish Time for the swarm is verified against number of peers present in the network with download rates ranging from 500, 1000 and 2000 Kbps in Figure 4. For a constant download rate, say 2000 Kbps, *the Download Finish Time increases linearly with number of peers.* This trend is attributed due to the fact that network load increases as more peers join the network, thereby increasing the download finish time. *In addition, when rate is doubled from 500 Kbps to 1000 Kbps, the Download Finish Time also doubles*, when viewed from the point of view that number of peers remains same, say 10.

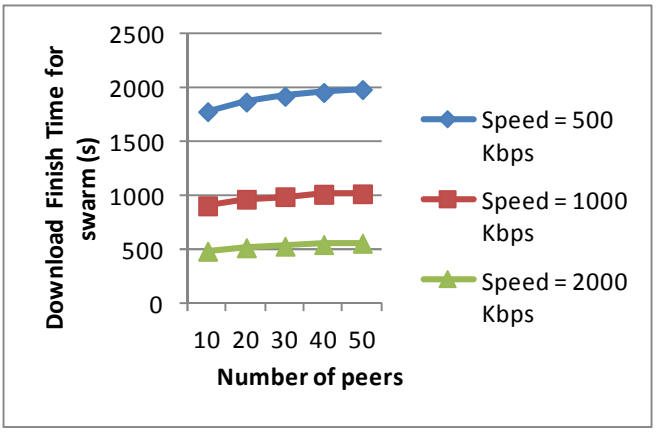

Figure 4: Number of peers vs. Download Finish Time

B. Real time analysis using LibTorrent library

Open source *libtorrent* library was compiled with qTorrent option as GUI enabled and its client example binary was used to download actual files over the Internet in order to observe the behavior of BitTorrent protocol in real world.

To measure the effect of factors affecting the performance efficiency, *libtorrent* library was compiled, in particular the an example program called *client_test*, which implements a complete BitTorrent client was built. Subsequently, it was used to download numerous files from Internet by passing the requisite torrent file via command line. This section presents a representative view of the observations made during the runs of such downloads. Care was taken to download with relevant parameters which affected the download rate of the file.
Particularly, the download performance was checked against two parameters:
i) Upload rate. The rate at which sub-pieces of a downloaded file is provided to other peers. The downloading rate was

checked against varying upload rates. Figure 5 shows the graph depicting the effect of upload rates on download rate of torrent file. Here, the download rate is taken as average. The average number of peers was taken as 50 within a swarm. Conclusion: *Interestingly, it was observed that increasing the upload speed increases the download speed as well, a verification of tit for tat strategy, however, after a certain upload rate threshold, it is observed that the download rate decreases.*

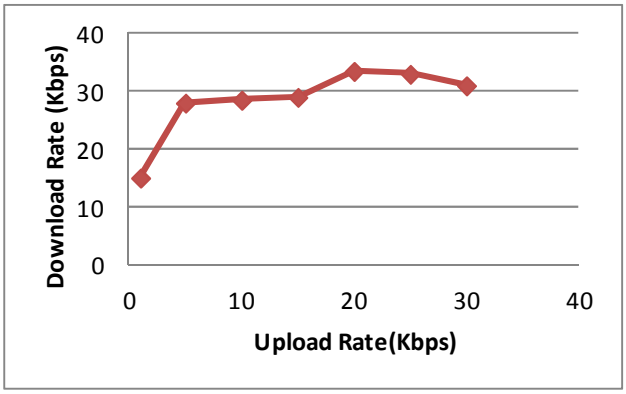

Figure 5: Upload Rate vs. Download Rate

ii) Number of peers connected, also known as swarm size. The peers in connection selected by Tracker, form a group that helps in downloading file to each other. The downloading speed was checked under different swarm size. Here also, the download rate is taken as average. Figure 6 shows the graph depicting this table. Conclusion: *Although, number of peers does help in download rate considerably due to sharing of file pieces, however, beyond a point, it results in decrease in download rate (speed).*

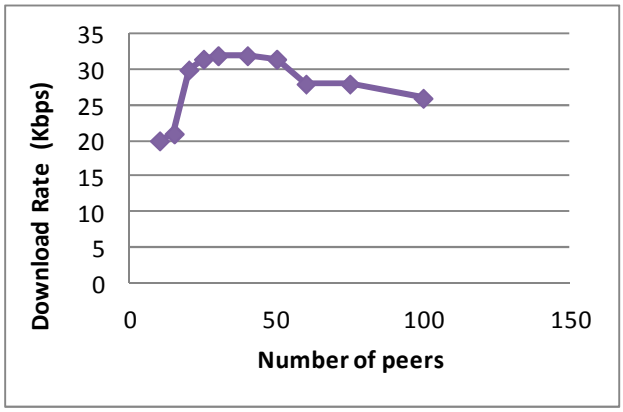

Figure 6: Number of Peers vs. Download Rate

VI. CONCLUSION

We studied in this paper the working of BitTorrent Protocol and various mechanisms used to achieve optimal performance by the protocol. In this paper we investigated the impact of number of peers on download rate through NS-2 simulator and *libtorrent* client. It was found that the Total Download Time decreases when number of peers increase due to an increased availability of file pieces across multiple peers. However, Download Time increases beyond certain limit due to increase of traffic in the network.

Another inference drawn from NS-2 based analysis is that howsoever late a peer joins the network, it finishes in almost the same time as compared to peers joining early.

Based on *libtorrent* based analysis, it was found that download rate is maximum when the upload rate is configured neither too low nor too high. This practical trend is in compliance to the Tit-for-Tat strategy of BitTorrent protocol. In this regard it is further concluded that BitTorrent incentive mechanism is playing a key role in peer-to-peer downloads.

REFERENCES

[1] Cohen, Bram (2001-07-02). "BitTorrent — a new P2P app". Yahoo eGroups, 2001.

[2] Hendrik Schulze, Klaus Mochalski, “Internet Study 2008/2009”, Ipoque, 2009.

[3] "BitTorrent and µTorrent Software Surpass 150 Million User Milestone", Bittorrent.com, Jan., 2012.

[4] “BitTorrent Has More Users Than Netflix and Hulu Combined--and Doubled”, www.fastcompany.com., Jan., 2011.

[5] “Online Video Rankings”, www.comscore.com, Aug., 2010.

[6] Michal Kryczka, Rubén Cuevas, Carmen Guerrero, Arturo Azcorra, Angel Cuevas, “Measuring the BitTorrent Ecosystem: Techniques, Tips, and Tricks”, Communications Magazine IEEE

[7] B. Cohen, “Incentives build robustness in BitTorrent,” in First Workshop on Economics of Peer-to-peer Systems, Berkeley, USA, June 2003

[8] Raymond Lei Xia and Jogesh K. Muppala, Senior Member, IEEE, “A Survey of BitTorrent Performance”

[9] M. Izal, G. Urvoy-keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, “Dissecting BitTorrent: Five months in a torrents lifetime,” in Passive and Active Measurements (PAM), LNCS, vol. 3014, April 2004, pp. 1–11.

[10] A. Legout, G. Urvoy-Keller, and P. Michiardi, “Rarest first and choke algorithms are enough,” in IMC ’06: Proc. 6th ACM SIGCOMM conference on Internet measurement. New York, NY, USA: ACM, 2006, pp. 203–216.

[11] Chih-Lin HU* and Zong-Xian LU, “A Trace Study of BitTorrent P2P File distribution with Downloading-Side Performance Measurement and Analysis”, Advanced Communication Technology (ICACT), 2012 14th International Conference on, Feb. 2012, pp. 875-880

[12] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, “Analyzing and improving a BitTorrent networks performance mechanisms,” INFOCOM 2006. 25th IEEE International Conference on Computer Communications Proceedings, pp. 1–12, April 2006.

[13] P. A. Felber and E. W. Biersack, “Self-scaling networks for content distribution,” in Int. Workshop on Self-* Properties in Complex Information Systems, Bertinoro, Italy, May-June 2004

[14] G. Urvoy-Keller and P. Michiardi, “Impact of inner parameters and overlay structure on the performance of BitTorrent,” INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp. 1–6, April 2006

[15] “NS2 Simulator” http://www.isi.edu/nsnam/ns/

[16] Kolja Eger, Tobias Hoßfeld, Andreas Binzenh¨ ofer, Gerald Kunzmann, “Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations”, Proceedings of the ACM SIGCOMM second workshop on Use of P2P, GRID and agents for the development of content networks, pp. 9-16