

A Trace Study of BitTorrent P2P File Distribution with Downloading-Side Performance Measurement and Analysis

Chih-Lin HU* and Zong-Xian LU

Department of Communication Engineering, National Central University, Taoyuan, Taiwan 32001, R.O.C.
clhu@ieee.org; h741234@gmail.com

Abstract—The peer-to-peer networking technologies and systems provide a rapid and scalable content distribution mechanism for the increasing client population to access an explosive volume of media files on the Internet. Whereas the BitTorrent protocol and its derivatives are the most popular peer-to-peer file sharing applications, this paper proposes the performance measurement and analysis of BitTorrent protocols with an extensive volume of real trace logs. This examination reveals many new characteristics and information into the virtue of BitTorrent protocols in terms of downloading-side metrics, including overall download time and download bandwidth utilization, which have not been well specified in prior works that mainly focused on system-oriented and uploading-side performance measurements.

Index Terms—BitTorrent, P2P, Internet measurement.

I. INTRODUCTION

As we witness the astounding increases of Internet media content and client population in the global Internet, it is crucial to have a fast, efficient, and reliable approach to quicken the delivery of Internet content, reduce media file downloading time, and avoid the possibility of server overload. Because traditional client-server and content distribution network (CDN) systems easily fall into the problems of scalability and performance degradation, the peer-to-peer (P2P) networking and systems [1][2] have emerged as a new distributed computing paradigm for content distribution applications on the Internet.

Regarding several differences among client-server and content distribution network (CDN), and P2P paradigms, including peer duality, service capacity, peer churn and overlay organization, the intrinsic difference is peer duality: each peer plays a dual role of both a service server and a client of the implemented service [3]. By contrast, traditional client-server and CDN paradigms primarily count on one or a smaller number of media content providers as servers that are responsible for a huge number of clients. Both paradigms can lead to the problems of scalability and performance degradation due to limited service capability, bandwidth, and throughput, while request workload increases substantially by a growing client population. Although adding more servers can alleviate such critical situations, the cost to upgrade back-end infrastructures and systems is very expensive with extra maintenance and management. Comparatively, P2P paradigm is not associated with any dedicated servers. In peer-to-peer systems, a peer requests media content or resource from its

peers, and also stores and serves those to its peers. Hence, increasing peer population not only increases request workload, but also produces a concomitant increase in service capacity to process the workload [4]. Peers can leverage huge low-cost service capabilities such as storage, transmission throughput and computation power among peers in networks to achieve scaled performance in a collective manner.

When P2P file sharing applications, such as Napster, Gnutella, eDonkey, and BitTorrent, produce a major portion of today's Internet traffic, the BitTorrent and its derivatives indeed contribute a dominant fraction of Internet traffic [5]. Considering the popularity of BitTorrent Protocols [5] thus invokes our study of performance measurement upon BitTorrent-initiated P2P traffic workloads. Our literature survey learned that most of prior studies [6][7][8][9] focused on analyzing the interaction between BitTorrent trackers and clients, the availability of file segments scattered in the networks, and the efficiency of uploading peer selection methods [10][11]. However, the above analyses mainly account for the system-oriented or uploading-side performance measurements. To our best knowledge, little information was available from the client-oriented or downloading-side concerns, such as file downloading time [12], the number of inter-peer connections, and bandwidth utilization at the receiver side.

The goal of this paper is to provide complementary information of BitTorrent performance study from the downloading-side dimension. We deploy an array of BitTorrent clients on the Internet and conduct thousands of practical experiments to record TCP I/O data logs at the downloading sides. With extensive collections of real traffic data, we define several downloading-side metrics, including the size of downloaded media file, downloading-side bandwidth capacity and number of transmission connections, which are used to investigate the sensitivities of downloading-side performance. Accordingly, with extensive collections of real trace data, our scrutiny finds new characteristics and insights into the virtue of BitTorrent protocols, which have not been noted in prior works.

The rest of this article is organized as follows. Section II introduces BitTorrent protocols and reviews some related works on BitTorrent's performance measurement. Section III describes the setting of experimental environments and specifies several definitions of performance metrics. Section IV presents the performance results and discussions. Several concluding remarks are provided in Section V.

II. BACKGROUND KNOWLEDGE AND RELATED WORK

A. BitTorrent Protocols

As Fig. 1 depicts, a BitTorrent system consists of three roles, trackers, torrent servers and peers, supporting basic operations of creating, publishing and downloading torrents, and sharing files. A peer treats the file as a number of pieces of identical size between 32 KB and 4 MB (typically, 256KB). To share files, a peer first creates a *torrent* that contains file meta-data, e.g., names, lengths, piece size used and piece-based hashing codes for data integrity, and a tracker's URL. Torrents are typically published on websites, and are registered with at least one tracker. The tracker records all the peers that have either partial or complete files, so it dictates peers to connect with others for file sharing. When a peer wants to download a file, it first obtains a torrent from some torrent server. The peer connects to the tracker indicated in the torrent, from which it receives a random list of peers currently downloading pieces (in the size of 16KB) of the same file. The peer then makes parallel TCP connects to those peers to obtain various file segments and gradually completes the file. After that, this peer will also start to upload pieces of the file to other peers.

BitTorrent systems apply several intrinsic policies below, which all BitTorrent clients must abide by to maintain file distribution, bandwidth utilization, and service fairness.

- Local Rarest-first (LRF) policy: To improve the overall system throughput, each peer in the BitTorrent system tries to share the file segments which are the rarest in its neighborhood to enhance the availability of file segments in the network under *flash-crowd* scenarios [7]
- Tit-for-tat (TFT) policy: To perform reciprocal incentives, the TFT policy penalizes the free riders that purely download but do not upload files by reducing their downloading capacities. This allows peers with more uploading contribution to be rewarded with faster downloading speed and reduced downloading time, thereby ensuring service fairness in the system.
- Optimistic unchoking (OU) policy: A peer generally chooses the peer of the highest transmission bandwidth or speed to download any file segment and change the unchoking states with other peers (4 peers by default). To explore other peers that may have higher bandwidth, the OU policy is used by any peer to try unused connections to learn whether there are alternatives that might be better than those currently used ones (per 30 seconds). Thus, a peer can still unchoke and exchange segments with neighboring peers regardless of its remaining capacity

B. Related Work

The work in [8] examined the infrastructure of BitTorrent file sharing systems, including Web servers/mirrors for directory services, meta-data distribution, and content sharing services. This study addressed four aspects: the availability of mirror servers, torrent file server and trackers, the integrity of both content itself and associated meta-data, the flash-crowd handling, and the download performance. As examined, the assumptions of Poisson arrival and departure processes for

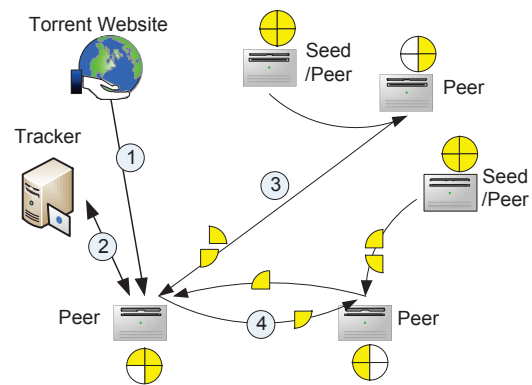


Fig. 1. The steps of BitTorrent traffic measurement.

downloaders and seeds, which are used in [9], is contradicted by the results in [8]. It turns out that 90% of the peers had a download speed below 520 kbps; the average download speed of 240 kbps allowed peers to fetch even large files in one day. An important observation is the power-law relation between the average download speed and the number of downloads at that speed. When the popularity drops and the last peer/seed with certain content goes off-line, the content dies soon.

The work in [7] found several limitations of BitTorrent systems. First, for a new file published by 45 to 55 hours, its download speed will reach the maximum, after which service availability becomes poor quickly due to the exponentially decreasing peer arrival rate in reality. Second, client performance is unstable, and fluctuates widely with the peer population. Third, existing systems could provide unfair services to peers, where peers with high downloading speed tend to download more and upload less.

The simulation-based study in [6] was conducted at several metrics, including peer link utilization, file download time, and fairness among peers. The results confirmed that BitTorrent performs near-optimally in terms of uplink bandwidth utilization and download time, except under certain extreme conditions such as node populations with heterogeneous bandwidths and the origin server with scarce bandwidth. Several findings were included in [6]. First, the TFT policy fails to prevent unfairness across nodes in terms of volume of content served. This unfairness arises when high bandwidth peers connect to low bandwidth ones. Second, a seed node should serve unique blocks at first to ensure diversity in the network instead of duplicate blocks that can be performed well by the leechers. Third, the LRF policy is critical in ensuring that new leechers quickly have something to offer to other nodes.

Our study differs from most of measurement works that focused on system-oriented service capability, availability, uploading efficiency, and fairness among peers. We think the download duration time till a file is downloaded completely, that is a distinct factor which users generally mind at the downloading side. We analyze real trace data and provide beneficial information to understand the unclear relationship between download duration time and the number of peers that a client peer could ever make connections with. Hence, this study can be complementary to many previous research efforts.

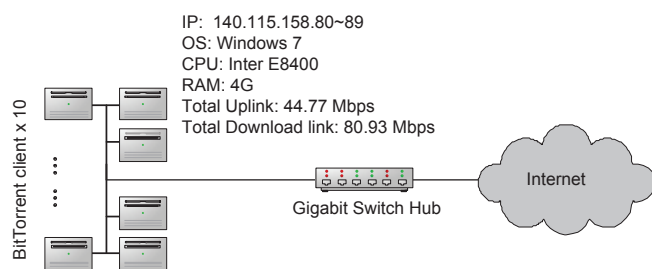


Fig. 2. An environment setting with 10 client peers.

III. SYSTEM DEPLOYMENT

This section describes the deployment of BitTorrent client applications for real trace generation, the processes of collecting trace data and data cleaning on logs, and the measurement metrics used to derive the numerical information.

A. Environment Setting

Fig. 2 depicts an in-house environment where contains 10 BitTorrent peer clients [13] on separate PCs that run with common hardware specifications and Windows 7 operating systems. All end hosts were assigned static IP addresses and wired to a Gigabit switch hub connecting to the Internet. For BitTorrent traffic measurement, we consider two media types, video and music, with different file sizes. Given with some torrent files, as listed in Table 1, we conducted BitTorrent peer clients to download media files and, meanwhile, dumped all TCP/IP I/O traffic data using the Wireshark network packet analyzer attached to the same hub in the same network domain.

B. Measurement Steps

In reference to Section II-A, the process of downloading a file involves a swarm of uploading peers and a responsible tracker indicated by the torrent that the client peer owns. The following describes the measurement steps, as Fig. 1 shows.

- 1) With a torrent published by a website, the downloading peer parses this torrent and resolves specific meta-data to have the responsible tracker URL and file attributes for data integrity in the media delivery.
- 2) The downloading peer connects to the tracker and will be responded with a random list of candidate uploading peers that have a partial or complete file.
- 3) The downloading peer repeatedly communicates with some candidates to exchange BitTorrent-specific information, such as version and maps of available file segments. By the LRF policy, the peer downloads distinct file segments simultaneously from other peers.
- 4) Meanwhile, the peer shares downloaded segments with other peers under reciprocity principles in systems. The peer will become a source peer of this file, i.e., seed node, after it completely downloads the file.

C. Log Processing

With a massive volume of raw TCP traffic data, the log process runs in three phases to extract relevant data. The first is to

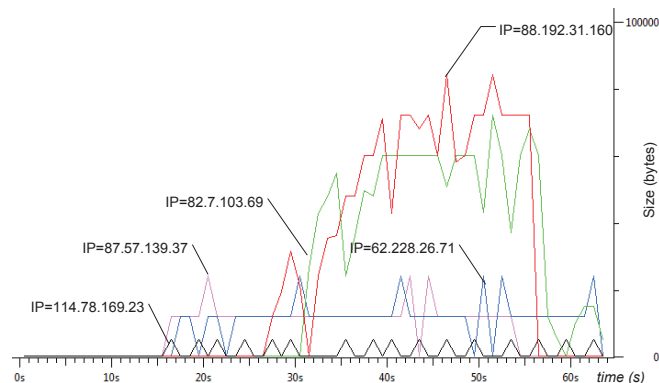


Fig. 3. Log processing: an illustrative example in a real sample.

distill BitTorrent-related data using the Wireshark-customized filter "bittorrent.piece.data" to filter out non-BitTorrent traffic data. The second is to determine any TCP transmission session associated with a pair of network endpoints (IP addresses and port numbers) during file downloading. The final phase sorts all transmission sessions in chronological order, and pieces up the total duration with a number of transmission intervals corresponding to distinct file segments of the same file.

The log process results in both temporal information and network location information during downloading a file. For example, we can know that how many uploading peers have contributed to this downloading of a file, and that where the uploading peers are. Therefore, we can know that how many peers have been selected from the list to be connected with, and further determine that how many uploading peers and their associated connections are active in a time scale. The above information is used for the analysis of downloading-side bandwidth allocation, utilization, and download time duration.

Since downloading a file involves a number of discontinuous transmission sessions in a time scale, the log process needs to concatenate all time pieces to have the overall download time. Given with a chronological sequence of transmission sessions pertaining to the same file, the overall download time of a file is calculated from the moment of the beginning of the first session to the end of the last session. Then, the number of active connections and peers at each time moment can be further calculated during downloading the file. For instance, Fig. 3 depicts a real sample of I/O data graph where a peer communicates with five uploading peers for different segments of the same file. The file download took about 48s, which started at 15.5s (the first peer via 87.57.139.37) and finished up at 63.5s (the last peer via 82.7.103.69). It can be seen that there are three active peers/connections at time 25s and five at time 50s. Hence, upon bandwidth allocation and the maximal quota of peers that a peer is permitted to use, it is possible to examine bandwidth utilization and efficiency of peer selection at the receiver side in BitTorrent networks.

D. Analytical Metrics

We conduct trace experiments with three downloading-side tuning parameters, i.e., file size, maximum of outward connections to uploading peers and total bandwidth download

TABLE I
A COLLECTION OF REAL TRACE LOGS WITH TORRENT INFORMATION.

	Small file 1	Large file 1	Large file 2	Large 3
File name	04 Firework.mp3	[OPFansMaplesnow][One_Piece] [509][848x480].rmvb	[jumpcn][naruto] [444][Big5][848x480].rmvb	[CASO&SumiSora][Hanasakuiroha] [22][GB][RV10].rmvb
File type	audio	video	video	video
File size	6993.92 KB	98611.2 KB	113164 KB	107776 KB
Upload link	256 KB/s	8192 KB/s	8192 KB/s	8192 KB/s
Download link	128, 256, 512, 1024 KB/s	128, 256, 512, 1024 KB/s	128, 256, 512, 1024 KB/s	128, 256, 512, 1024 KB/s
Trace duration	2011/1/20–2/20	2011/8/10–13	2011/8/22–24	2011/8/29–31
Times	10	5	5	5

capacity, to produce real trace data. With such traffic logs, we can derive three performance criteria, i.e., overall download time, average download bandwidth and average bandwidth utilization, which can affect the overall download time.

- *Overall download time*: Given with a chronological sequence of transmission sessions for downloading a particular file, the calculation of overall download time begins from the moment which the first transmission session starts to the moment which the last session completes the downloading of that file segment.
- *Average Download Bandwidth*: The download bandwidth is given with the file size divided by the overall download time. The measurement runs each experimental case in five times and then obtains the average.
- *Average Bandwidth utilization*: The bandwidth utilization (in percentage) is given with the download bandwidth divided by the total download bandwidth. The measurement runs each experimental case in five times and averages the values of bandwidth utilizations.
- *Maximal number of outward connections* (to uploading peers) that a client peer can simultaneously make to download different file segments. This parameter will be used to examine the influence of this parameter on the download time and bandwidth utilization.
- *Total download bandwidth* that a client can use aggregately since a client peer can make multiple outward connections in chorus. When the aggregation of download throughput is limited, we will observe the descent of overall download time by the number of uploading peers.
- *File size*: Depending on various media types, i.e., audio and video, it takes various times to download media files of different sizes. Under a set of download bandwidth capacities and peer numbers, we will observe the cost-effectiveness as tuning these two factors to improve overall download time and bandwidth utilization.

IV. MEASUREMENT RESULTS

This section reports the results of BitTorrent traffic measurement at the downloading side. As Fig. 2 shows, we deploy Vuze BitTorrent client peers [13] on a group of end hosts in a high-speed LAN linked to the Internet. Each end host runs the NetLimiter software to control download bandwidth and data throughput, which are more than the limit of total download bandwidth as assigned in advance of downloading any file.

Table I lists the information of media file, torrents, and a variety of parameter values that are used to generate the

real trace logs. Particularly, the measurement setting considers two basic parameters, i.e., file size and total download bandwidth capacity, which can directly affect the overall download time. Furthermore, under a certain bandwidth allocation, the measurement setting further examines the average download bandwidth and bandwidth utilization in response to the incremental value of the maximal number of outward connections to uploading peers. Thus, the real traffic collection contains extensive data logs with thousands of downloading cases in different file sizes, download bandwidth capacities and maximum of outward connections to uploading peers, respectively given with the value ranges of [music <10 MB, video >100 MB], [128, 256, 512, 1024 KB/s] and [1, 2, ..., 32], for the performance measurement in this section. In what follows, we report the performance results, observations and discussions on real traffic traces.

A. Results with Small Files (Music Type)

The measurement results in Fig. 4 show the average values by 10 times in the case of downloading small files with file sizes about 7 MB. Obviously, the overall download time decreases in response to the increase of uploading peers, namely, the number of outward connections, since a client peer can make only one connection to any uploading peer at one time by default. As shown in Fig. 4(a), as the maximal number of peers is smaller than 6, the download time reduces sharply by adding one more uploading peer. This descending curve turns out to be smoother after the maximal number of uploading peers is larger than 7 and more. This is because the initial list of uploading peer candidates, which the tracker offers in the beginning of downloading files, is *raw* without order by uploading bandwidth capacity. The client peer can only passively depend on the BitTorrent-initiated TFT and OU policies to find better substitutes of uploading peers in order to improve the downloading performance.

On the other hand, there might be an argument that a client peer can repeatedly request a list of uploading peer candidates and then attempt to replace the slow uploading peers. However, from the viewpoint of bandwidth utilization, it is found that this effect is not obvious in this case of a smaller file. As Fig. 4(c) shows the result of bandwidth utilization, the curves in the cases of higher download bandwidths, i.e., 512 and 1024 KB/s, go up very slightly; particularly, the percent difference is smaller than 10 % on average as the number of uploading peers rises from 10 to 32. The finding in trace data of downloading smaller files shows that when

the file is downloaded completely, the number of actually-used outward connections never reaches the maximum as the maximum is set higher (>15). Thus, it is learned that the excess of bandwidth allocation or transmission throughput is not of cost-effectiveness for reducing the overall download time as downloading small files.

B. Results with Large Files (Video Type)

In order to investigate the effects by increasing the maximum of uploading peers, this measurement conducted thousands of downloading cases for three distinct large files (>100 MB) that were launched into the BitTorrent networks in August 2011. For each large file as listed in Table I, it took about 3~4 days to finish all downloading cases in 5 times to have the average results, as depicted in Figs. 5, 6 and 7. There are several interesting findings below.

First, as comparing Figs. 5(a), 6(a) and 7(a) with Fig. 4(a), the descending curves are similar as the maximum of uploading peers increases from 1 to 6, which can be explained with the aforementioned reasons. Second, when the maximum of uploading peers is more than 6, the bandwidth utilization rises gradually by increasing more and more uploading peers. This observation means that, after repeatedly asking for an updated list of uploading peer candidates, the client peer can communicate with more uploading peers with larger uploading capacities. The results in Figs. 5(b), 6(b) and 7(b) confirm this observations that the average download bandwidth of all concurrent connections rises as the maximum of uploading peers increases. Third, the check of trace logs found that only few cases have the number of actually-used concurrent connections equal to the maximum. Due to the nature of flash-crowding or peer churning issues [14], it is difficult for P2P system to always guarantee the service capacity and availability [7]. Thus, a client peer cannot reach 100 % bandwidth utilization in practice. Fourth, generally speaking, peers have the asymmetric bandwidth capacity since the uploading bandwidth or transmission throughput is lower than the downloading one. When a client peer is assigned a high limit of total download bandwidth, the uploading peers may not meet the limit oppositely over the Internet. For example, as shown in all Figs. 5(b), 6(b) and 7(b), the curves of high download bandwidth with 512 and 1024 KB/s are lower than their upper limits, respectively. Correspondingly, as shown in Figs. 5(c), 6(c) and 7(c), their average bandwidth utilizations are near or lower than 70 % in most cases with various numbers of uploading peers. Notice that the results in Figs. 7(b) and (c) are much low due to the possibility of lower file popularity. Finally, the other cases with low bandwidth allocations, i.e., 128 and 256 KB/s, can obtain very high bandwidth utilization, close to full utilization, while more uploading peers are connected in the course of downloading large files. This observation also implies the issue of cost-effectiveness between download bandwidth and download time as similar as the mention in the case of small files. The results in Figs. 6(a) and 7(a) support this implication, in evidence, since the cases with either low or high bandwidth allocation can achieve similar download time when the maximum of uploading peers increases.

V. CONCLUSION

This paper has accounted for the investigation of BitTorrent P2P file sharing systems in terms of downloading-side bandwidth allocation and bandwidth usage. With several downloading-side parameters, i.e., total bandwidth allocation, maximum of concurrent connections to uploading peers and file size, we conducted thousands of measurement cases to download different files from the Internet. Upon the collection of real trace logs, we examined BitTorrent protocols against download time and download bandwidth utilization, to figure out performance efficiency and cost-effectiveness. Accordingly, we provide new information and insights into the influences of download bandwidth and connections to uploading peers, which can be useful for performance improvement at the downloading side. The effort of this study can be a complement to tracker- or uploading-side performance measurements and analysis on BitTorrent systems.

REFERENCES

- [1] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, December 2004.
- [2] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, March 2008.
- [3] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proceedings of IEEE INFOCOM'03*, vol. 3, April 2003, pp. 2188–2198.
- [4] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *Proceedings of IEEE INFOCOM'04*, vol. 4, March 2004, pp. 2242–2252.
- [5] ipoque GmbH, "Internet study 2008/2009," 2009, <http://www.ipoque.com/resources/internet-studies/internet-study-2008-2009>.
- [6] A. R. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving a bittorrent network's performance mechanisms," in *Proceedings of IEEE INFOCOM'06*, April 2006.
- [7] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, October 2005.
- [8] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, February 2005.
- [9] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proceedings of the ACM SIGCOMM'04*, August 2004.
- [10] C.-L. Hu, Y.-H. Chang, D.-Y. Chen, and Y.-W. Chen, "Simple peer selection strategies for fast and fair peer-to-peer file sharing," in *Proceedings of the 12th International Conference on Advanced Communication Technologies (ICACT'10)*, February 2010, pp. 208–213.
- [11] C.-L. Hu and D.-Y. Chen, "Distributed pairing for file sharing in large-scale peer-to-peer networks," in *Proceedings of the 13th International Conference on Advanced Communication Technologies (ICACT'11)*, February 2011, pp. 492–497.
- [12] C.-J. Wu, C.-Y. Li, and J.-M. Ho, "Improving the download time of bittorrent-like systems," in *Proceedings of IEEE ICC'07*, June 2007, pp. 1125–1129.
- [13] Vuze, Inc., "Vuze BitTorrent client software 4.6," 2011, <http://sourceforge.net/projects/azureus/files/vuze/>.
- [14] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, October 2006, pp. 189–202.

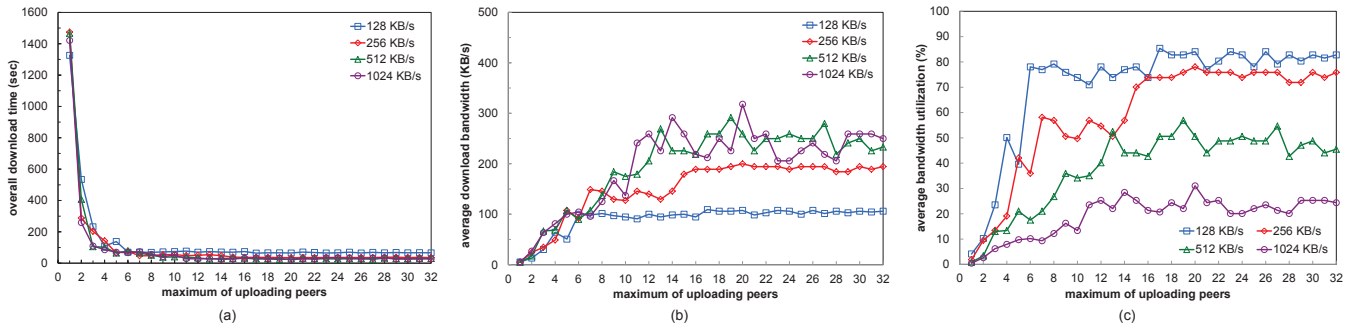


Fig. 4. Small file: (a) overall download time, (b) average download bandwidth and (c) average bandwidth utilization.

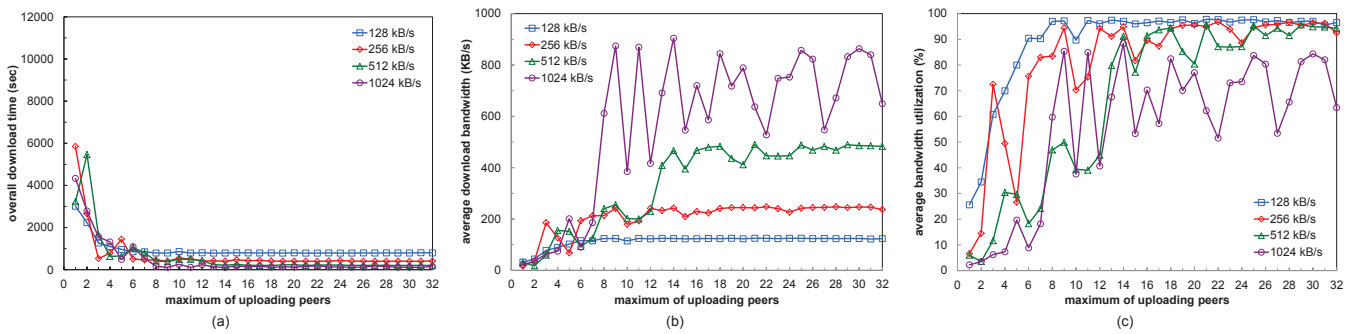


Fig. 5. Large file 1: (a) overall download time, (b) average download bandwidth and (c) average bandwidth utilization.

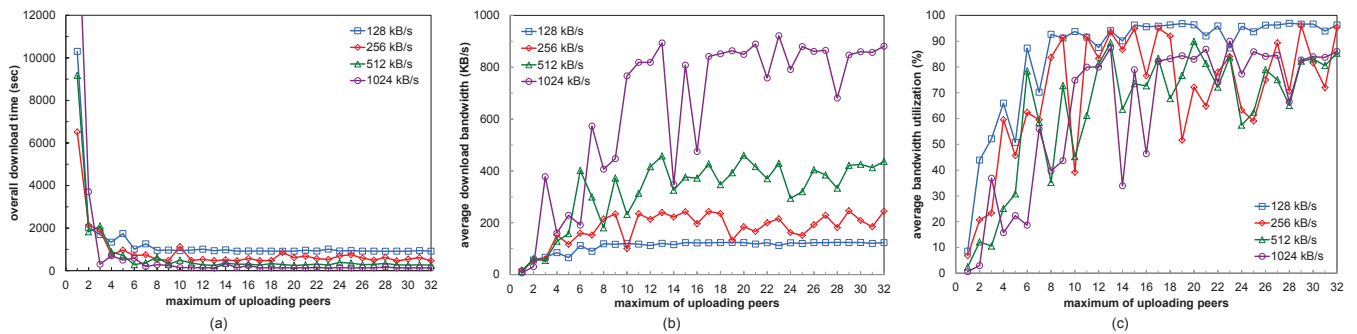


Fig. 6. Large file 2: (a) overall download time, (b) average download bandwidth and (c) average bandwidth utilization.

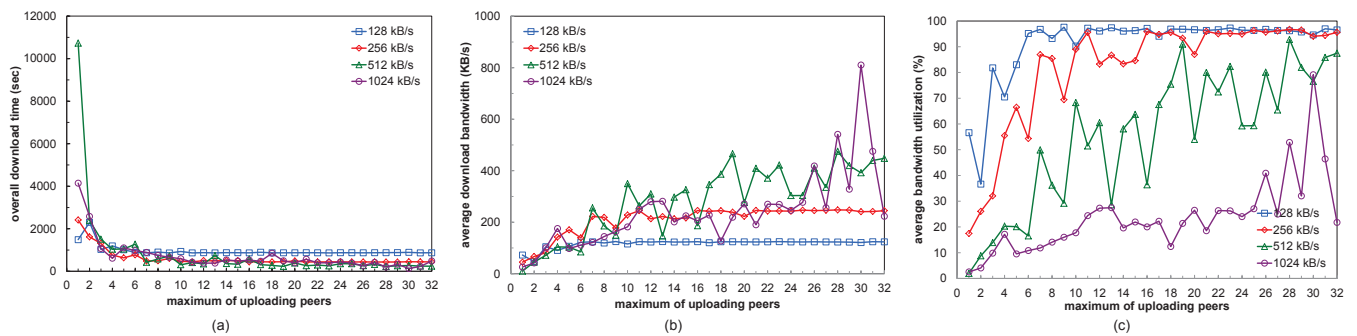


Fig. 7. Large file 3: (a) overall download time, (b) average download bandwidth and (c) average bandwidth utilization.