PAPER Special Section on The Internet Architectures, Protocols, and Applications for Diversified Futures

# Deep Inspection of Unreachable BitTorrent Swarms

**Masahiro YOSHIDA**[†a)], *Nonmember and* **Akihiro NAKAO**[†b)], *Member*

**SUMMARY** BitTorrent is one of the most popular P2P file sharing applications worldwide. Each BitTorrent network is called a swarm, and millions of peers may join multiple swarms. However, there are many unreachable peers (NATed (network address translated), firewalled, or inactive at the time of measurement) in each swarm; hence, existing techniques can only measure a part of all the peers in a swarm. In this paper, we propose an improved measurement method for BitTorrent swarms that include many unreachable peers. In essence, NATed peers and those behind firewalls are found by allowing them to connect to our crawlers by actively advertising our crawlers' addresses. Evaluation results show that the proposed method increases the number of unique contacted peers by 112% compared to the conventional method. Moreover, the proposed method increases the total volume of downloaded pieces by 66%. We investigate the sampling bias among the proposed and conventional methods, and we find that different measurement methods yield significantly different results.

***key words:*** *peer-to-peer networks, BitTorrent, network measurement, unreachable peers*

## 1. Introduction

In recent years, BitTorrent has become the most popular P2P networks throughout the Internet. Due to a growing number of video sharing websites such as YouTube, BitTorrent traffic is steadily increasing year by year [1]. As BitTorrent becomes popular, many researchers, companies and copyright holders devote their efforts to understanding and characterizing BitTorrent networks. Especially, the accurate measurement of BitTorrent networks, i.e., the minimization of sampling biases in BitTorrent measurements, can benefit studies on BitTorrent in the following ways. It may lead to a better understanding of the topology and characteristics of BitTorrent *swarms*, a collection of peers that participate in the distribution of a specific file through a BitTorrent application. Accurate swarm information is valuable for understanding and improving the performance of the BitTorrent protocol and traffic control methods such as P4P [2], Ono [3] and so on. However, several issues continue to hinder the accurate measurement of BitTorrent swarms.

First, there are many unreachable peers in each swarm and unreachable peers causes a serious sampling bias in *peer-level measurement methods*, i.e., when a crawler tries to contact each peer to obtain detailed information [4]–[8]. Peer-level measurement methods can obtain information such as IP address, service port number, latency, download rate, connection status, available pieces of the content (*piece-bitmap*), and neighboring peer addresses. However, each swarm includes many unreachable peers (NATed, firewalled, or inactive at the time of the measurement). Owing to their inability to connect the crawler to unreachable peers, conventional peer-level measurement studies can measure only a part of all the peers in the swarm [9].

Second, the peer-level measurement method may degrade the performance of BitTorrent systems. In order to contact peers, the crawler has to contact *trackers* in advance. A tracker is a server that keeps track of which peers are in the swarm. Peers report information to the tracker periodically and, in exchange, receive information about other peers to which they can connect. However, each tracker retains crawler addresses for a long time, and peers may obtain inactive crawler addresses until the trackers discard them following the measurement. When peers try to contact the inactive crawler, they cannot download their desired files and this may slow down file completion. We must measure swarms with a minimal impact on the BitTorrent performance.

In this paper, we propose an improved measurement method for unreachable peers with minimal impact on the BitTorrent performance. In developing our method, we focus on a BitTorrent peer that simultaneously works as a client and server to the other peers in the swarm. In the conventional peer-level measurement method, the crawler does not work as a server and disregards all incoming accesses for unreachable peers. To improve connectivity between the crawler and unreachable peers, our crawler opens service ports and accepts all incoming accesses from peers to retrieve detailed information. Moreover, to increase the availability of files, our crawler aggressively advertises reachable peer addresses to the target swarm.

There are four contributions of this paper, as follows: First, through our extensive study of peers' connectivity included in the collected information, we find that, on average, 8.16% of peers are reachable in each swarm. This is because most peers are NATed or behind a firewall. Further, we investigate how long each tracker retains inactive peer addresses. We show that 99.11% of trackers keep inactive peer addresses for at least 30 minutes. This out-of-date information directly affects the peer-level measurements.

Second, we develop an improved peer-level measure-

Manuscript received June 6, 2012.
Manuscript revised September 28, 2012.

[†]The authors are with Applied Computer Science Course, Interfaculty Initiative in Information Studies, Graudate School of Interdisciplinary Information Studies, The University of Tokyo, Tokyo, 113–0033 Japan.
a) E-mail: yoshida@nakao-lab.org
b) E-mail: nakao@iii.u-tokyo.ac.jp

DOI: 10.1587/transinf.E96.D.249

ment method. Although unreachable peers do not accept a connection from our crawler, they may access our crawler if they know its address. For this reason, our crawler aggressively advertises its address to the target swarm and accepts all incoming accesses. Our method increases the number of unique peers contacted by 112.75% compared with the conventional methods.

Third, to achieve our measurement purpose, our crawler collects peers' incoming connections. However, our crawler does not have any pieces of the file, and this may slow down file completion. To avoid this problem, our crawler advertises the set of reachable peer addresses to incoming peers. Because our crawler cooperates by advertising reachable peer addresses to the target swarm, our method increases the total volume of downloaded pieces by more than 66.07% compared to the case without our cooperation.

Last, to evaluate the impact of different measurement methods on peer-to-peer (P2P) characterization, we compare the characteristics of the target P2P network measured concurrently using different methods, such as churn [11] and peer's degree distribution. We show that the results exhibit significant differences according to the measurement methods, and that our method captures the most detailed information of all, revealing that the other methods may result in false characterization of P2P networks due to a lack of detail.

To the best of our knowledge, this is the first study into peer-level measurement that considers both the unreachable peers problem and the performance degradation problem. Our main contribution in this paper is that our scrutiny of P2P characteristics not only reinforces the sampling bias inherent in the existing approaches, but also suggests the possibility of improving the efficiency of P2P file sharing applications.

The remainder of the paper is organized as follows. In Sect. 2, we briefly explain the BitTorrent protocol. In Sect. 3, we discuss related work on BitTorrent measurements, and we expose the problems of the peer-level measurements in Sect. 4. Section 5 introduces our proposed measurement method, and and Sect. 6 explains the method employed by the implemented crawler. In Sect. 7, we show the effectiveness of our measurement method. In Sect. 8, we compare the measurement results obtained with different measurement methods. We then discuss the key implications of our findings on the design of the BitTorrent system in Sect. 9, and present our conclusions in Sect. 10.

## 2. BitTorrent Protocol

BitTorrent is a P2P file distribution application that achieves efficient file sharing through so-called *swarming* where each file to be shared is divided into small fragments and the peers that intend to download the file form the same overlay network (the *swarm*) and exchange the fragments to reconstruct the file among the swarm.

A swarm consists of peers, a *tracker* and a *.torrent file*. Peers are divided into two kinds, seeders, and leechers. A peer who does not have the complete file is called a *leecher*, whereas a peer that has the complete file is called a *seeder*. For each swarm, a server is created that is known as a tracker. The tracker does not host any files, but provides a peer discovery service to the swarm. It contains all the peer addresses of the swarm and returns a random subset of peer addresses (average 50) per inquiry. A peer joining a swarm asks the tracker for a random subset of active peer addresses, and then exchanges file fragments with those random peers. Note that a single tracker may manage multiple swarms simultaneously.

A .torrent file contains metadata on the file to be shared and the tracker (e.g., the name, the identifier, the size, tracker addresses, and piece hashes). The .torrent file is usually hosted on a web server. Peers obtain the location of the .torrent file corresponding to the file to be downloaded, and download the .torrent file to obtain the address of the tracker to contact.

## 3. Related Work

### 3.1 Tracker-Level Measurements

A tracker enables coordination among peers in order to distribute their files in the BitTorrent swarm. The *tracker-level measurement method* is defined as to obtain statistics regarding swarms by contacting trackers but not peers.

The tracker holds the IP address and service port number of each peer in a swarm, and it is this information that is used by tracker-level measurement studies. Such studies may be classified into two groups. The first group uses log traces from trackers [10]–[15]. A BitTorrent tracker program is set up on PC servers to contact active peers in the tracked swarm. This method can only obtain the complete set of active peer addresses when the swarm is tracked by one tracker. However, as shown in our previous study [16], only 10% of swarms are managed by a single tracker, and the rest are managed by multiple trackers. Because peers may not need to contact all trackers for the download to complete, this approach may overlook some peers in the swarm. Moreover, log traces are often problematic to obtain as they require agreement from content providers. The second group uses crawling (or spidering) techniques [8], [9], [16], [18], [20]. In order to obtain a comprehensive picture of the entire set of BitTorrent swarms, a crawler periodically contacts each tracker to obtain all the peer addresses that have been tracked. This method allows us to obtain the peer addresses of the entire BitTorrent swarm within a short time. However, this method requires tens of PC servers and high-speed internet links to conduct a large-scale measurement, such as that in [17]. Each approach is effective for addressing particular needs; however, there are limits to what we can study through peer addresses.

### 3.2 Peer-Level Measurements

There are only a few existing approaches for measuring BitTorrent swarms using peer-level measurement methods [4]–[8]. These methods can obtain information including IP address, service port number, latency, download rate, connection status, available pieces of the content (piece-bitmap), and neighboring peer addresses. Siganos et al. study the deviant behavior of BitTorrent peers [4]. With an instrumented BitTorrent client, they analyze the top 600 swarms on The Pirate Bay for 45 days. Piatek et al. [5] also use instrumented BitTorrent clients in developing their reputation system. They obtain two datasets, from 13,353 swarms and 55,523 swarms. Although this is the simplest method, it is difficult to measure a large number of BitTorrent swarms. One of the reasons is that the measurement process is slowed down by lots of the client features, such as GUI, synchronous sockets, and inefficient threadpools. In addition, as there are over 56 variants of the original BitTorrent application [21], different types may exhibit different measurement results. Another measurement platform is developed by Pouwelse et al. on 100 nodes of the ASCI Supercomputer [6]. They aggressively contact known peers of 108 swarms for two weeks and study the download speed of peers, flashcrowds, content lifetime, and so on. Likewise, Kryczka et al. [7] collect the graph topology of 250 real torrents through PEX messages [22]. However, these measurements cannot contact unreachable peers because they do not accept all incoming accesses from peers. Therefore, their results are only valid for reachable peers. Iosup et al. [8] extend this to contact unreachable peers in each swarm. They develop the multiprove framework for correlated measurements, which simultaneously conducts active and passive measurements to contact unreachable peers. They measure 2,000 swarms for one week and study connectivity, geolocation, RTT (round-trip time) of peers, and so on. From the number of empirical BitTorrent measurements, few [8] even consider aspects of the unreachable peer problem. These can only contact a fraction of the unreachable peers, and their datasets still exhibit a sampling bias (we explain this in Sect. 5).

## 4. Problems with Peer-Level Measurement

We have identified three problems. First, each swarm contains many unreachable peers. In our previous work [16], we measure BitTorrent swarms through our tracker-level measurement method. We identify about 4.4 million unique swarms and 10 million unique peers in a day. To measure the connectivity of each peer, our crawler tries to contact all known peers in each swarm. Figure 1 shows the ratio of reachable peers in each swarm. We can see that 32.21% of swarms consist of only unreachable peers, and this means that their swarms are inactive. Only 20% of peers are reachable in 93.97% of swarms, and only in 2.89% of swarms are over 50% of peers reachable. The average ratio of reachable peers in each swarm is just 8.16%. This is because most peers are NATed or behind a firewall. In peer-level measurements, the crawler needs to contact peers to retrieve detailed information. However, the large proportion of unreachable peers would cause a serious sampling bias.

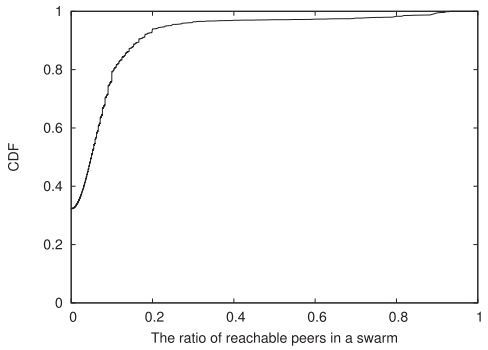


**Fig. 1** Ratio of reachable peers in each swarm. Horizontal axis is the ratio of reachable peers in each of the 4.4 million unique swarms. Vertical axis is the CDF. About 94% of swarms are composed of less than 20% of reachable peers. The average ratio of reachable peers in each swarm is about 8%.

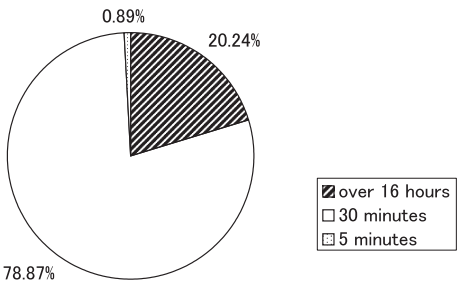


**Fig. 2** Holding time of inactive peer addresses in each active tracker. 0.89% of trackers hold inactive peer addresses under 5 minutes. However, 99.1% of trackers retain inactive peer addresses for at least 30 minutes.

Second, we investigate how long each tracker retains inactive peer addresses. Another reason for the high number of unreachable peers in a swarm is that trackers advertise inactive peer addresses. We identify about 49,000 unique trackers, but only 499 of these are active. Our crawler contacts each active tracker and reports its own address as an active peer. Our crawler then becomes inactive immediately. Figure 2 shows the holding time of inactive peer addresses in each active tracker. We find that 99.11% of trackers retain inactive peer addresses for at least 30 minutes. Moreover, 20.24% of trackers keep inactive peer addresses for over 16 h. Because tracker-level measurement studies use a dataset that includes many inactive peer addresses, it would be difficult to get a complete picture of BitTorrent swarms from these measurement studies. Moreover, with peer-level measurement, the crawler needs to retrieve peer addresses from trackers in order to contact peers in each swarm. For the same reason, the crawler would obtain inactive peer addresses and subsequently fail to contact peers. In particular, we should avoid collecting peer addresses from trackers that hold inactive peer addresses for over 30 minutes.

Finally, peer-level measurements may have the impact on download of pieces, thus, user's convenience. In general, BitTorrent crawlers do not have any file pieces and they are uncooperative to exchange files. However, trackers cannot distinguish crawlers from normal peers and trackers distribute crawler's addresses to swarms. Thus, peers misidentify the crawler's addresses as the seeder's (or leecher's) addresses. When peers try to contact the crawler, they cannot download their desired files and this may slow down file completion. We must measure swarms with a minimal impact on the BitTorrent performance.

## 5. Proposed Method for Peer-Level Measurement

In order to retrieve detailed information about the peers in a swarm, *a conventional peer-level measurement method*, whereby a crawler obtains peer addresses from trackers and attempts to contact each peer, is often used to measure as many peers as possible [4], [6], [7]. However, this method disregards all incoming accesses and cannot contact many unreachable peers. For this reason, the obtained dataset may contain a large sampling bias. In order to contact unreachable peers, Iosup et al. [8] have conducted passive measurements where a crawler reports its own address to a tracker and waits to be contacted by other peers. This method can retrieve detailed information about unreachable peers when they try to connect to the crawler. However, this method can usually only contact a portion of the unreachable peers. This is because the crawler address is slowly diffused to the whole swarm and most unreachable peers cannot obtain the crawler address. Each peer contacts a tracker every 15 minutes to receive information about other peers to whom they can connect, and the tracker offers a random subset of peer addresses per single inquiry (50-200 addresses each time). For this reason, the crawler has to wait a long time to be contacted by unreachable peers, and these peers may leave the swarm during the measurement period. This has a big impact especially on large swarms.

We believe that peer-level measurement must contact unreachable peers for a short time. In order to achieve this, our crawler opens many service ports and reports these addresses to a tracker. These addresses have the same IP address but different service port numbers. The tracker regards our crawler addresses as many NATed peer addresses. This method increases the probability of crawler addresses being included in the random subset. Peers can easily obtain our crawler addresses when they contact the tracker, and they would then try to contact our crawler to download desired pieces of a file. As a result, our method can contact many unreachable peers in a short time, and we can thus minimize the sampling bias of the obtained dataset. An overview of our peer-level measurement scheme is as follows (see Fig. 3):

1. In order to obtain peer addresses to contact, our crawler repeatedly collects a random subset of peer addresses until no more new peers can be discovered from a tracker. To avoid collecting many inactive peer addresses, our crawler does not contact trackers that hold inactive peer addresses for over 30 minutes (see Fig. 2). When our crawler contacts the tracker, it also reports its own address as a seeder in order to attract leechers. Because our crawler opens many service ports, it reports the same IP address but a different port number each time until the tracker obtains all of the crawler addresses. Following this process, the tracker is tracking many crawler addresses.
2. When a peer has no addresses from which to download desired pieces, they try to contact the tracker to obtain a new random subset of peer addresses. Because the tracker holds many crawler addresses, the probability of the random subset including at least one crawler address is high. Moreover, BitTorrent defines the *PEX protocol* [22], which allows each peer to directly exchange addresses that they can connect to. Due to the PEX protocol, peers can also obtain our crawler addresses even if they do not contact the tracker during the measurement period. As a result, our method can efficiently diffuse our crawler address to many unreachable peers.
3. After obtaining peer addresses from the tracker, our crawler attempts to contact them to retrieve detailed information. In addition, our crawler accepts all incoming connections from peers through its opened service ports. When our crawler establishes a connection with a peer, it sends a subset of 10 crawler addresses via a PEX message. To conclude our measurement after a fixed period of time, our crawler stops contacting peers when we can obtain all of a leecher's piece-bitmap, which must be obtained in order to analyze the download ratio of each peer in a swarm. When the seeder has all of the valid pieces of a file, the seeder's piece-bitmap consists entirely of "1". In contrast, leechers have a wide variety of piece-bitmaps because they have different incomplete files. By contacting the tracker, we can obtain the number of leechers and the number of seeders in the target swarm. When the number of piece-bitmaps obtained from leechers equals the num-

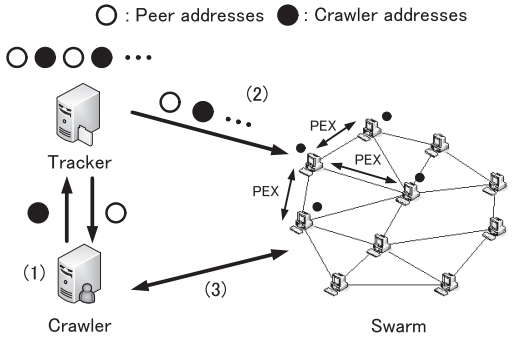


**Fig. 3** Proposed peer-level measurement method. Our crawler aggressively diffuses its own address to the target swarm, allowing unreachable peers to connect to our crawler.

ber of leechers in a swarm, we can regard the other peers as seeders. Accordingly, we assume that all other peers' piece-bitmaps are "1," and we conclude our measurement within minutes.

Our crawler aggressively advertises its address to NATed peers and waits for them to connect. When this happens, the crawler can obtain their addresses. However, each tracker retains inactive addresses for up to 30 min, and so peers may obtain inactive crawler addresses until they are discarded by the trackers after the end of the measurement period. Our crawler does not have any file pieces, which may slow down file completion process. To avoid this problem, our crawler sends the subset of reachable peer addresses to incoming peers after the end of the measurement. Because reachable peer addresses are well diffused to the target swarm, our method increases the file availability after the end of measurement. Our crawler sends random subsets of 10 reachable peer addresses via PEX messages.

## 6. Implementation

In our attempt to obtain detailed information from as many peers as possible, we have implemented a high-performance crawler. Our crawler collects detailed information about peers in a systematic and automated manner through the method described in Sect. 5. Our crawler can open any number of service ports (1-65,536) and accepts all incoming accesses from peers. Because about 25% of incoming packets are encrypted, we also implement BitTorrent protocol encryption [23] to decode them. Our crawler is developed on a single PC server (CPU: Intel Core i7 2.8 GHz, Memory: 16 GB, HDD: 3 TB). The PC server is connected to the internet via a 100 Mbps access link. The crawler is developed in C# and runs on Windows XP 64-bit edition. To effectively use the access link, our crawler uses asynchronous I/O completion ports (IOCP) and can support more than 10,000 concurrent T CP connections. In addition, we change the socket-related registries (MaxUserPort, TCPMaxHalfOpen, ForwardBufferMemory, and so on) in the Windows OS.

Some BitTorrent applications support the IPv6 addressing protocol [21] and BitTorrent trackers support IPv6 peers [24]. Thanks to tracker's extended messages [24], our crawler can identify IPv4 peers and IPv6 peers. When our crawler sends the subset of reachable peer addresses, our crawler can select the appropriate addressing protocols for each peer. For this reason, our proposed method also supports IPv6 peers.

## 7. The Effectiveness of Our Measurement Method

### 7.1 Evaluation Setup

We have evaluated the effectiveness of our method over an internet environment. At the beginning of our measurements, the target swarm is composed of over 10,000 peers (seeders: 5,845; leechers: 4,639; file size: 351 MB). Four trackers track this swarm in parallel. Our crawler obtains peer addresses from, and reports its address to, these four trackers. To compare the proposed method with conventional peer-level measurement methods, we set up three crawlers to measure the target swarm. We use PCs with the same specification for each crawler. The three crawlers are connected to the Internet via the same access link, but their IP addresses are different (not NATed). The *first crawler* uses only the *client* function, such as [4], [6]. It tries to contact each peer but does not accept any incoming connections. The *second crawler* uses the client function and the *server* function, such as [8]. It accepts all the incoming connections, but trackers hold only one crawler address. The *third crawler* uses the proposed method (see Sect. 5). This crawler accepts all the incoming connections and the trackers hold 1,000 crawler addresses. After the measurements are done, the third crawler waits for 30 minutes for incoming accesses in order to advertise reachable peers. These three crawlers measure the target swarm concurrently. We measure the same swarm three times with the same settings.

Note that our proposed method works for not only for large swarms, but also for small swarms. Actually, the swarm size of our evaluation is over 10,000 peers but 95% of swarms are composed of fewer than 100 peers. In general, trackers return a random subset of peer addresses (on average 50) per inquiry for peers. In small swarms, the probability of crawler addresses being included in the random subset is higher than in large swarms. Therefore, our proposed method works with the smaller number of crawler's addresses and we can complete measurements with a shorter time. Accordingly, our proposed method is more effective for small swarms rather than large swarms.

### 7.2 Evaluation Result

In this section, we demonstrate the effectiveness of our proposed method. Although the target swarm of our measurement is large, our method can obtain all of a leecher's piece-bitmap in 6 minutes.

#### 7.2.1 The Total Number of Incoming Accesses

First, we show how many peers attempt to contact our crawler. Our crawler collects peers' incoming connections in order to communicate with NATed peers and firewalls. Table 1 shows the total number of incoming accesses over a period of 6 minutes. When trackers hold only one crawler address (i.e., when using the second tracker), the total number of incoming accesses during the measurement period is less than ten. This is because the crawler address is slowly diffused to the whole swarm and most peers may not obtain the crawler address. In contrast, 4,102-12,462 peers contact our crawler when the trackers hold 1,000 crawler addresses. This shows that our method efficiently diffuses crawler addresses to the whole swarm and that our method successfully collects peers' incoming connections.

**Table 1** Total number of incoming accesses over a period of 6 minutes. Our crawler can collect many peers' incoming connections when we diffuse 1,000 crawler addresses to the target swarm.

| | 1 crawler address | 1,000 crawler addresses |
|---|---|---|
| Measurement 1 | 7 | 12,462 |
| Measurement 2 | 3 | 8,797 |
| Measurement 3 | 9 | 4,102 |

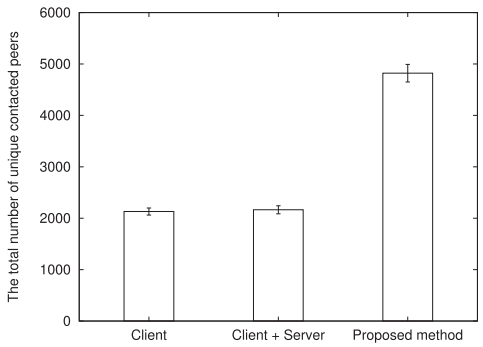


**Fig. 4** Total number of unique peers contacted over a period of 6 minutes. Vertical axis shows the total number of unique contacted peers. Our proposed method increases the number of unique peers contacted by about 113% compared to other methods.

### 7.2.2 The Total Number of Unique Peers Contacted

Next, we show how many peers we can contact. Figure 4 shows the total number of unique peers contacted over the 6-minute period. We contact 2,130 unique peers with the first crawler, which uses only the client function and does not accept any incoming connections. Because many peers are unreachable, the first crawler fails to connect to them. We can contact 2,164 unique peers using the second crawler. The second crawler opens one service port and accepts all the incoming packets. However, because only a small number of peers can obtain our crawler address, the second crawler does not allow us to contact many unreachable peers. However, our proposed method contacts 4,821 unique peers over the 6-minute period. Using our method, many unreachable peers can obtain our crawler addresses from trackers and PEX messages. These peers aggressively attempt to contact our crawler. Accordingly, our method increases the number of unique peers contacted by 112.75% compared to the conventional methods. From these results, our method efficiently increases the number of unique peers contacted, which can help us contact NATed peers and those behind firewalls.

### 7.2.3 The Cumulative Number of Unique Leechers' Piece-Bitmaps

Our measurements aim to obtain all of the leechers' piece-bitmaps. A piece-bitmap represents the pieces that have been successfully downloaded. Figure 5 shows the cumulative number of unique leechers' piece-bitmaps obtained. Because a leecher's piece-bitmap varies with time, our method must obtain the piece-bitmap in a short time. There are 4,639 unique leechers in the target swarm at the start of the measurement. Using the first crawler, we can obtain 2,712 leechers' piece-bitmaps in total, of which 2,076 of them are unique. With the second crawler, we obtain 2,714 leechers' piece-bitmaps in total, and we identify 2,068 of them as unique. Thus, with the conventional methods, we cannot obtain even half of the unique leechers' piece-bitmaps. However, our method obtains 17,693 piece-bitmaps in total, and we can obtain all of the unique leechers' piece-bitmaps within 6 minutes. These results demonstrate that our method can conduct sufficient peer-level measurements in a short time and reveal all of the peers' download rates.

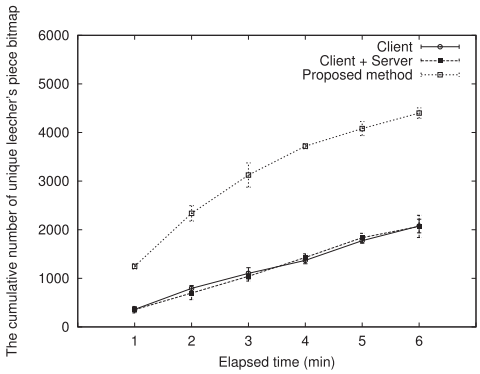


**Fig. 5** Cumulative number of unique leechers' piece-bitmaps. Horizontal axis is the elapsed time. Vertical axis is the cumulative number of unique leechers' piece-bitmaps collected over a period of 6 minutes. Our proposed method can obtain all piece-bitmaps within 6 minutes.

In practice, it is impossible to distinguish between these two cases. However, our method achieves this because a firewalled peer may contact our crawler. If the contact does not occur within a certain time frame, we may incorrectly mark this as an inactive peer. Although our method cannot solve this problem completely, we can reduce its impact by advertising a sufficient number of crawler addresses to the target swarm. For example, we advertise 1,000 crawler addresses to a target swarm that includes 10,000 peers. In this case, we can obtain all of the leechers' piece-bitmaps within 6 minutes. Because our crawler can advertise any number of crawler addresses to trackers, our method would be able to handle this problem for any swarm size.

### 7.2.4 The Total Volume of Downloaded Pieces

After the end of the measurements, the third crawler sends a subset of reachable peer addresses to incoming peers. This is because our crawler addresses remain in trackers for up to 30 minutes, which may interfere with peers' piece exchanges. We attempt to understand how effective it is to advertise reachable peers to the target swarm. Figure 6 shows the total volume of downloaded pieces after the end of the measurements. We estimate the total volume of downloaded pieces from the piece-bitmaps obtained from the leechers. Our crawler collects piece-bitmaps for 30 minutes after the end of measurements. When our crawler is non-cooperative

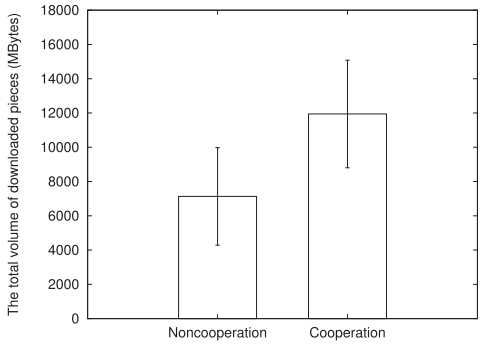


**Fig. 6** Total volume of pieces downloaded after the end of the measurements. Vertical axis shows the total volume of downloaded pieces in a 30 minutes period. When our proposed method is cooperative (i.e., our crawler sends reachable peer addresses to the target swarm), the download volume is increased by 66%.

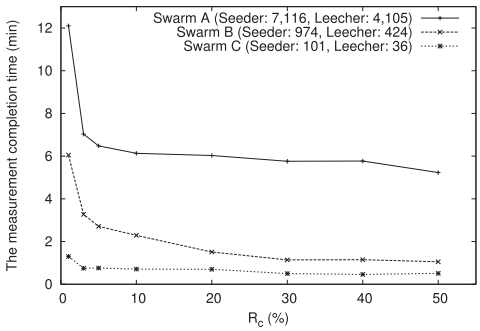


**Fig. 7** Measurement completion time with respect to the ratio of crawler addresses to all peer addresses. Horizontal axis is the ratio of crawler addresses to all peer addresses. Vertical axis shows the measurement completion time. The measurement completion time decreases rapidly until $R_c$ = 5% and is saturated above $R_c$ = 10% in all three swarms.

(i.e., does not send reachable peer addresses after the end of the measurement), the total volume of downloaded pieces is 7,131 MB. When our crawler is cooperative (i.e., does send reachable peer addresses after the end of the measurement), the total volume of downloaded pieces is 11,943 MB. This represents an increase in download volume of 66.07% by using reachable peer addresses. This means that advertising reachable peer addresses is an effective way to minimize the reduction of file availability because the swarm contains many unreachable peer addresses.

#### 7.2.5 The Ratio of Crawler Addresses To Peer Addresses

Next, we measure the effect of the number of crawler addresses advertised by our crawler. In our proposed method, the ratio of crawler addresses to all of the peer addresses is a key factor that directly affects the measurement completion time. Our proposed method completes its measurements when our crawler obtains all of the bitmaps of the file fragments from the leechers (each bitmap indicates which file fragments are held by the leechers). If we advertise many crawler addresses to the target swarm, the ratio of crawler to peer addresses increases. Thus, there is a higher probability of peers obtaining our crawler addresses through trackers and PEX messages from neighboring peers. Therefore, firewalled and NATed peers can quickly obtain our crawler addresses and attempt to contact our crawler. We measure the completion time with respect to changes in the ratio of crawlers to peers for three cases with different swarm sizes. $n_c$ denotes the number of crawlers and $n_p$ denotes that of peers in the swarm to be measured. The ratio of crawlers to peers $R_c$ is calculated by $R_c = \frac{n_c}{n_p}$. Figure 7 shows the measurement completion time when we change $R_c$. Our results show that the measurement completion time decreases with $R_c$. In particular, the measurement completion time decreases rapidly until $R_c$ = 5%, at which point the rate of decrease slows down. Although our crawler can advertise any number of addresses to a given swarm, too many crawler addresses would overload trackers and peers and may decrease the performance of the swarm. Furthermore, the measurement completion time is saturated beyond $R_c$ = 10% for each of the three swarms measured. For this reason, we select $R_c$ = 10% as an appropriate value.

## 8. Comparison of Measurement Results Obtained from Different Methods

To understand the impact of different measurement methods on P2P characterizations, we compare the characteristics of the target P2P network measured concurrently using different methods. Especially,we evaluate the churn [11] and peer's degree distribution.

We use four measurement methods to measure the target swarm. The first crawling method contacts only trackers; thus, it is defined as a tracker-level measurement. The second crawling method contacts peers via the client function; thus, it is defined as a peer-level measurement (denoted as Client). The third crawling method contacts peers via the client function and the server function; thus, it is defined as a peer-level measurement (denoted as Client + Server). This method opens only one service port. The fourth crawling method is our method (see Sect. 5), and it is defined as peer-level measurement (denoted as Proposed method). These four crawlers measure the same target swarm concurrently.

We measure one swarm (seeders: 310; leechers: 209; file size: 683 MB) with a granularity of seconds to analyze the churn. We then measure one swarm (seeders: 10,166; leechers: 8,323; file size: 1.29 GB) with a granularity of minutes to analyze the peer's degree.

### 8.1 Churn Characteristics

To understand the churn characteristics, the session length is the most important property [11]. The session length indicates how long peers remain in the network. Figure 8 shows the distribution of session length for different measurement methods. In the tracker-level measurement, we measure the session length of peers in the list by recording the elapsed time between joining and leaving the tracker. In the peer-level measurement, our crawler periodically contacts each

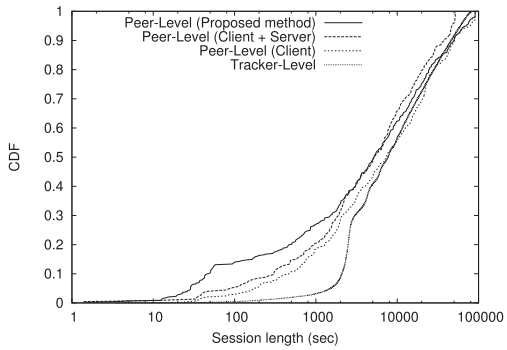


**Fig. 8** Comparison of the session length. Horizontal axis is the session length of each peer. Vertical axis is the CDF. Four crawler types measure the same target swarm concurrently. In the tracker-level measurement, we cannot estimate the session length for short-lived peers because most trackers keep short-lived peers' addresses for at least 1,800 s. However, our proposed method succeeds in contacting many short-lived peers before they leave the swarm.

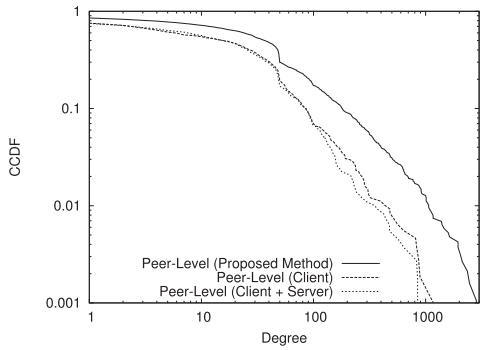


**Fig. 9** Comparison of the degree distributions. Horizontal axis is the peer's degree. Vertical axis is the complementary CDF. Three crawler types measure the same target swarm concurrently. The degree distribution only becomes large with our proposed method. This is because our proposed method can obtain many more PEX messages than other methods.

peer. We record all such events and define the session length as the duration between the first and last such event. We can see that the measured session lengths are extremely different between the tracker-level measurement and the peer-level measurement. In the tracker-level measurement, the distribution of session length rises sharply at 1,800 seconds. This is because most trackers keep the IP addresses of short-lived peers for at least 1,800 seconds. Trackers perform periodic clean-ups by removing from their lists IP addresses for which they have not received any updates or keep-alives. For this reason, the tracker-level measurement has difficulty in estimating the session length of short-lived peers. In contrast, peer-level measurement methods can estimate the session length of short-lived peers. The peer-level measurement method directly contacts each active peer and can thus identify the real session length without being affected by the tracker's retention of IP addresses. As a result, peer-level measurements are effective for estimating short session lengths.

However, we can also see a difference in the session length distribution among the three types of peer-level measurements. Notably, our method obtains very short-lived session lengths, which indicates that 14% of peers leave the swarm within 1 minute, whereas the Client + Server method reveals only 4% of peers leaving the swarm in this period. Likewise, the Client method shows 2% of peers leaving the swarm within 1 minute. This is because the Client + Server and Client methods cannot contact sufficiently many short-lived peers. In contrast, our method significantly increases the number of contacted peers compared with other methods (e.g., Sect. 7.2.2). For this reason, our method succeeds in contacting many short-lived peers before they leave the swarm.

### 8.2 Degree Characteristics

In the study of network properties, the degree of a peer in a swarm is the number of connections it has to other peers. We can obtain neighboring peer addresses for each peer through PEX messages. Note that our crawler addresses are removed from each PEX message. Figure 9 shows the degree distribution for the different measurement methods. The tracker-level measurement method cannot obtain degree information. Therefore, Fig. 9 only plots the degree distribution for the peer-level measurement methods. The degree distribution only differs slightly between the Client and Client + Server methods. However, there is a significant difference in the measured degree distribution between our method and the other methods. Using our method, 55% of peers had a degree of fewer than 30 and the average degree is 90, whereas 70% of peers had a degree of fewer than 30 with the other methods. The average degree is 39 with the Client method and 36 with the Client + Server method. Although we measure the same target swarm concurrently with these three methods, the degree only become large with our method. This is because our method can obtain many more PEX messages than the other methods. In the current BitTorrent protocol, one PEX message does not exhibit all the neighboring peer addresses for a peer, but instead includes only part of the neighboring peer addresses. Therefore, we need to collect many PEX messages from each peer to obtain the degree information. Moreover, our method can obtain PEX messages from unreachable peers. Hence, our method increases the number of PEX messages obtained by a factor of six compared to other methods.

The complementary CDF of the degree distribution drops off considerably at degree 50. This corresponds to the default degree of the popular BitTorrent client "$\mu$Torrent," which is used by 26% of peers.

## 9. Discussion

In this section, we discuss the possibility of applying our findings to improving the efficiency of P2P file-sharing systems. In particular, we focus on improving file availability by advertising reachable peer addresses to a swarm. In Sect. 7.2.4, our method is shown to aggressively diffuse reachable peer addresses to improve the connectivity

of peers. This leads to an increase of approximately 70% in download performance. This result implies that the connectivity of peers would significantly affect file availability in the BitTorrent system. Therefore, to maintain good connectivity for each peer, the BitTorrent system must be able to efficiently handle the significant fraction of unreachable peers (see Sect. 4).

We suggest three strategies to improve the connectivity of peers in a swarm. First, the BitTorrent system should enable NAT traversal for peers in each swarm. NAT traversal techniques allow unreachable peers to contact each other. There are many NAT traversal techniques for Internet applications [25], and most of them require assistance from a server at a publicly-routable IP address. We believe that, with a few modifications, a BitTorrent tracker can work as this server. Moreover, Guha et al. [26] develop STUNT (Simple Traversal of UDP Through NATs and TCP too) library, which extends STUN to include TCP functionality. BitTorrent client programs would use the library with a few modifications, and they would communicate through the existing NAT infrastructure without sacrificing the benefits of TCP. In addition, Lai et al. [27] implement their NAT traversal technique on BitTorrent. Thanks to their efforts, we can easily achieve our first strategy for improving the connectivity of unreachable peers.

Second, trackers should discard inactive peer addresses after a shorter period of time. To avoid contamination of the swarm by inactive peer addresses, it is essential to perform more frequent clean-ups by removing inactive peer addresses from trackers. More specifically, peers should use DHT (distributed hash table)-trackers instead of normal trackers. DHT-trackers can efficiently handle the churn, meaning that they discard inactive peer addresses after a minimal time. However, the current DHT-trackers cannot work well due to unreachable peers [28]. Therefore, we should also enable NAT traversal for DHT-trackers.

Third, Wu et al. [22] observe that PEX can significantly reduce the download time for some swarms. We also confirm this trend in Sect. 7.2.4. Therefore, peers must be aggressively exchanging active peer addresses via PEX messages. When peers follow this behavior pattern, active peer addresses are well diffused to the whole swarm, increasing the file availability between peers.

Although there are many unreachable peers in not only BitTorrent but also other P2P-like distributed systems, our method is effective for other P2P-like distributed systems. More specifically, our method works if P2P-like distributed systems have a peer discovery service. In BitTorrent, trackers work as the peer discovery service and trackers distribute our crawler's IP addresses to unreachable peers. Because unreachable peers try to contact to our crawler, our method can contact many unreachable peers in a short time. Fortunately, most of other P2P-like distributed systems also have a peer discovery service. For example, Winny, that is, one of the most popular P2P file sharing networks in Japan, contains over 50% of unreachable peers. But Winny has the peer discovery service and we have conducted measurements on Winny through our proposed method [29]. We can easily distribute our crawler's addresses to all peers through a peer discovery service and this fact also means our proposed method is effective for most of P2P-like distributed systems.

## 10. Conclusion and Future Work

In this paper, we propose a peer-level measurement method for unreachable peers in a swarm, and we evaluate our method on a real BitTorrent swarm. The main contribution of this study is that our scrutiny of P2P characteristics not only reinforces the significance of the sampling bias in existing approaches but also suggests the possibility of improving the efficiency of P2P file sharing applications. In detail, the contributions of this paper are four-fold.

First, our measurement results indicate that the current BitTorrent system is composed of many NATed and firewalled peers and that each tracker retains inactive peers for a long time. Unfortunately, these factors mean that we can only measure a fraction of the peers in a swarm.

Second, to solve the unreachable peer problem, we develop a method that accepts all incoming accesses from unreachable peers, and we show that our method successfully increases the number of unique contacted peers by 112.75%, as compared to conventional methods. Finally, our crawler connects all the leechers in the swarm within six minutes and also reveals all peers' download rates.

Third, we advertise reachable peer addresses to avoid a slowdown in the file completion time by unreachable peers. Our result shows that advertising reachable peer addresses increases the total volume of downloaded pieces by 66.07%.

Finally, we measure the characteristics of the target P2P network concurrently using different methods, and we show that the results exhibit significant differences according to the method of measurement. Our proposed method captures the most detailed information of those tested, and reveals that the other methods are subject to significant sampling bias due to their relative lack of detail.

In the future, we plan to scale our measurements to a wider scope. We are currently performing geographically distributed crawling on PlanetLab and CoreLab [30]. In addition, we plan to apply our method to other P2P file sharing protocols besides BitTorrent.

## Acknowledgements

This research is partly supported by Grants-in-Aid for Scientific Research (KAKENHI), No. 21300020.

### References

[1] ipoque, Internet studies (Online), http://www.ipoque.com/resources/internet-studies/ (Ref. 2012.10.01).
[2] H. Xie, Y.R. Yang, A. Krishnamurty, Y. Liu, and A. Silberschatz, "P4P: Provider portal for applications," Proc. ACM SIGCOMM'08, 2008.

[3] D.R. Choffnes and F.E. Bustamante, "Taming the Torrent: A practical approach to reducing cross-isp traffic in peer-to-peer systems," Proc. ACM SIGCOMM'08, 2008.
[4] G. Siganos, J. Pujol, and P. Rodriguez, "Monitoring the Bittorrent monitors: A bird's eye view," Proc. PAM'09, 2009.
[5] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Andershon, "One hop reputations for peer to peer file sharing workloads," Proc. NSDI'08, 2008.
[6] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, "The BitTorrent p2p file-sharing system: Measurements and analysis," Proc. IPTPS'05, 2005.
[7] M. Kryczka, R. Cuevas, C. Guerrero, and A. Azcorra, "Unrevealing the structure of live BitTorrent Swarms: Methodology and analyis," Proc. IEEE P2P'11, 2011.
[8] A. Iosup, P. Garbacki, J. Pouwelse, and D. Epema, "Correlating topology and path characteristics of overlay networks and the Internet," Proc. CCGrid, 2006.
[9] B. Zhang, A. Iosup, J. Pouwelse, D. Epema, and H. Sips, "Sampling bias in BitTorrent measurements," Lect. Notes Comput. Sci., Euro-Par 2010 - Parallel Processing, vol.6271, pp.484–496, 2010.
[10] M. Izal, G. Uroy-Keller, E. Biersack, P.A. Felber, A.A. Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in torrent's lifetime," Springer, Passive and Active Network Measurement, vol.3015, pp.1–11, 2004.
[11] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," Proc. 6th ACM SIGCOMM conference on Internet measurement, 2006.
[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," IEEE J. Sel. Areas Commun., vol.25, no.1, pp.155–169, 2007.
[13] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of BitTorrent-like systems," Proc. ACM SIGCOMM IMC'05, 2005.
[14] B. Zhang, A. Iosup, J. Pouwelse, and D. Epema, "Identifying, analyzing, and modeling flashcrowds in BitTorrent," Proc. IEEE P2P'11, 2011.
[15] A. Bellissimo, B.N. Levine, and P. Shenoy, "Exploring the use of BitTorrent as the basis for a large trace repository, "University of Massachusetts Technical Report, pp.04–41, 2004.
[16] M. Yoshida and A. Nakao, "A resource-efficient method for crawling swarm information in multiple BitTorrent networks," Proc. AHSP2011, 2011.
[17] C. Zhang, P. Dunghel, D. Wu, and K. Ross, "Unraveling the BitTorrent ecosystem," IEEE Trans. Parallel Distrib. Syst., vol.22, no.7, pp.1164–1177, 2010.
[18] G. Dan and G. Carlsson, "Dynamic swarm management for improved BitTorrent performance," Proc. IPTPS'09, 2009.
[19] S.L. Blond, A. Legout, F. Lefessant, W. Dabbous, and M.A. Kaafar, "Spying the world from your Laptop - Identifying and profiling content providers and big downloaders in BitTorrent," Proc. 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2010.
[20] N. Andrade, E. Santos-Neto, F.V. Brasileiro, and M. Ripeanu, "Resource demand and supply in BitTorrent content-sharing communities," Elsevier, Computer Networks, pp.515–527, 2009.
[21] Wikipedia, Comparison of BitTorrent clients (Online), http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_clients (Ref. 2012.10.01).
[22] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K.W. Ross, "Understanding peer exchange in BitTorrent systems," Proc. IEEE P2P'10, 2010.
[23] Wikipedia, BitTorrent protocol encryption (Online), http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption (Ref. 2012.10.01).
[24] BitTorrent.org, IPv6 Tracker Extension (Online), http://www.bittorrent.org/beps/bep_0007.html (Ref. 2012.10.01).
[25] Wikipedia, NAT traversal (Online), http://en.wikipedia.org/wiki/NAT_traversal (Ref. 2012.10.01).
[26] CMLAB, Implementing NAT traversal on BitTorrent (Online), http://www.cmlab.csie.ntu.edu.tw/franklai/NATBT.pdf (Ref. 2012.10.01).
[27] S. Guha and P. Francis. "An end-middle-end approach to connection establishment," Proc. SIGCOMM 2007, 2007.
[28] R. Jimenez, F. Osmani, and B. Knutsson, "Connectivity properties of mainline BitTorrent DHT nodes," IEEE P2P'09, 2009.
[29] M. Yoshida, S. Ohzahata, A. Nakao, and K. Kawashima, "Controlling file distribution in winny network through index poisoning," Proc. 23rd of International Conference on Information Networking, 2009.
[30] A. Nakao, R. Ozaki, and Y. Nishida, "CoreLab: An Emerging Network testbed employing hosted virtual machine monitor," Proc. ACM CoNEXT ROADS Workshop, 2008.

**Masahiro Yoshida** received B.S. degree (2008) and the M.S. degree (2010) in Engineering from Tokyo University of Agriculture and Technology, Japan. He is currently the Ph.D. candidate of the University of Tokyo, Japan. He has also been a research fellowship for young scientists at Japan Society for the Promotion of Science (JSPS) since 2011. His research interests include network measurement, P2P traffic control and network visualization.

**Akihiro Nakao** received B.S. (1991) in Physics, M.E. (1994) in Information Engineering from the University of Tokyo. He was at IBM Yamato Laboratory/at Tokyo Research Laboratory/at IBM Texas Austin from 1994 till 2005. He received M.S. (2001) and Ph.D. (2005) in Computer Science from Princeton University. He has been teaching as an Associate Professor in Applied Computer Science, at Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, the University of Tokyo since 2005. (He has also been an expert visiting scholar/a project leader at National Institute of Information and Communications Technology (NICT) since 2007.).