Supplementary File of the TPDS Manuscript TPDS-2012-05-0447-R1:

Challenges, Designs and Performances of Large-scale Open-P2SP Content Distribution

Zhenhua Li ^{1,2,3}, Yan Huang ^{2}, Gang Liu ^{2}, Fuchen Wang ^{2}, Yunhao Liu ^{4}, Zhi-Li Zhang ^{3}, Yafei Dai ^{1}

^{1} Peking University, Beijing, China, lzh@net.pku.edu.cn
^{2} Tencent Research, Shanghai, China, galehuang@tencent.com
^{3} University of Minnesota, Minneapolis, MN, US, zhzhang@cs.umn.edu
^{4} Tsinghua University, Beijing, China, liu@cse.ust.hk

Abstract—This supplementary file contains the supporting materials of the TPDS manuscript TPDS-2012-05-0447-R1 titled **"Challenges, Designs and Performances of Large-scale Open-P2SP Content Distribution"**. It improves the solidity and completeness of the TPDS manuscript.

1 Background: Server-based and Peer-to-peer Content Distribution

Content distribution on today's Internet operates primarily in two modes: (a) *server-based* and (b) *peer-to-peer* (P2P). On one hand, we have *server-based* content distribution that relies on Internet servers such as conventional web servers, CDN servers, and more recently, servers within massive data centers that are located closer to the "core" Internet. Examples of such systems include most commercial systems like Google, YouTube and Amazon. On the other hand, we have *P2P* content distribution that relies on a large number of user controlled end-hosts (e.g., home PCs, laptops or hand-held mobile devices) that are located on the "edge" of the Internet and collectively form a *peer swarm* [1], [2].

Both modes have unique characteristics and accompanying disadvantages. While server-based content distribution allows content providers better control over content distribution, it nonetheless requires significant upstart infrastructure and continuing operation costs (in terms of not only server and storage capacities, but also network bandwidth). In contrast, content distribution via dynamically and spontaneously formed peer data swarms incurs nearly zero cost, and is inherently far more scalable. However, due to each peer's limited capacity and dynamic nature (i.e., peers dynamically join or leave), the efficiency of P2P content distribution can be very poor and unpredictable.

2 Hardware Composition

The QQXuanfeng system is composed of four major building blocks: 1) *Content Index DB*, 2) *Content Crawler*, 3) *Content Validator* and 4) *Data Scheduler*, utilizing 53 commodity servers in total. The detailed hardware composition is listed in Table 1 (in Page 2). Note that all the Memory, CPU, Storage and Bandwidth information refers to one server. Besides, we record the average CPU utilization and bandwidth load (Internet) per server of each building block in a typical day (i.e., Dec. 15, 2011) in Table 2. A notable observation from Table 2 is that the bandwidth load of the Content Validator (0.29 Gbps $\times 12$) is very close to its maximum Internet bandwidth (0.3 Gbps $\times 12$), indicating that the Content Validator has become a bottleneck of our system and we plan to upgrade it soon.

TABLE 2
Average CPU utilization and bandwidth load (Internet) per server in a typical day.

Building block	CPU utilization	Bandwidth load
Content Index DB	27%	0.11 Gbps
Content Crawler	7%	0.06 Gbps
Content Validator	5%	0.29 Gbps
Data Scheduler	25%	0.19 Gbps

3 Additional Measurement Results

3.1 Link Popularity Distribution

In Fig. 1 and Fig. 2 we plot the *link popularity* distribution of the indexed contents in QQXuanfeng, where *link popularity* denotes the number of links pointing to an identical content. We find that the link popularity distribution is highly skewed: the 2.82% of contents with a very high link popularity (i.e., $\in [1000, 1000000)$) account to 38.11% of links, while the 40.23% of contents with a very low link popularity (i.e., $\in [0, 10)$) account to less than 1% of links.

3.2 User Access Pattern

To understand the basic user access pattern, we measured the user access behaviors regarding to an unbiased

TABLE 1
Hardware composition of QQXuanfeng.

Building block	Number of servers	Memory	CPU (4 cores)	Storage	Bandwidth
Content Index DB	15	16 GB	Intel Xeon 5130 @2.00 GHz	320 GB	1 Gbps (Intranet)
Content Crawler	12	8 GB	Intel Xeon X3210 @2.13 GHz	150 GB	1 Gbps (Intranet), 0.3 Gbps (Internet)
Content Validator	12	8 GB	Intel Xeon X3210 @2.13 GHz	150 GB	1 Gbps (Intranet), 0.3 Gbps (Internet)
Data Scheduler	14	8 GB	Intel Xeon X3210 @2.13 GHz	150 GB	1 Gbps (Intranet), 0.3 Gbps (Internet)

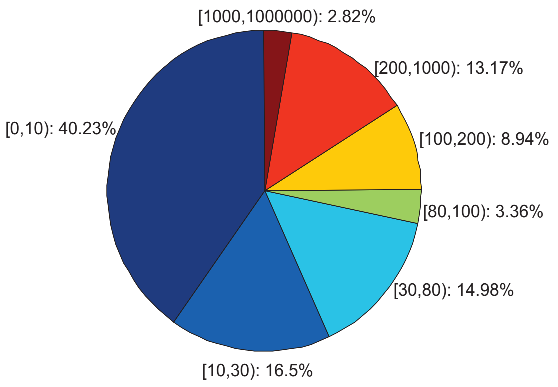


Fig. 1. Ratios of contents corresponding to various link popularities.

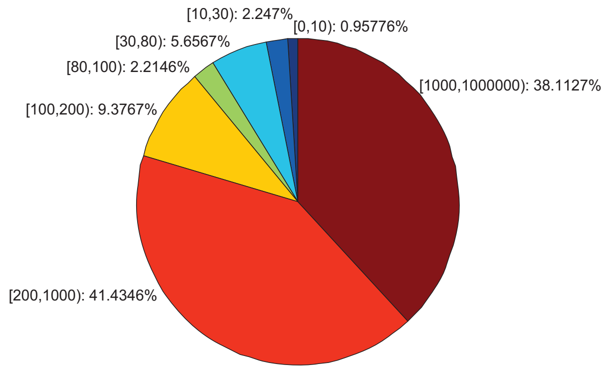


Fig. 2. Ratios of (content) links corresponding to various link popularities.

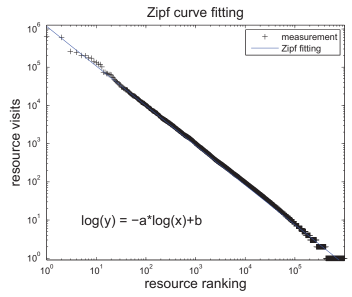
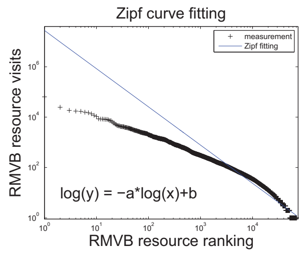


Fig. 3. User access pattern of all files well matches the Zipf distribution.



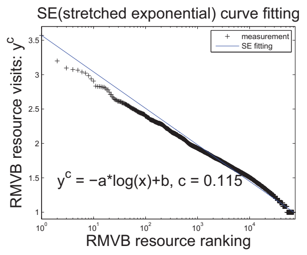


Fig. 4. User access pattern of .rmvb files deviates greatly from the Zipf distribution.

Fig. 5. User access pattern of .rmvb files approximates the SE distribution.

sample of files in QQXuanfeng in 4 days. In this sample, around 1 million files are visited for over 22 million times, where the average file size is 35.67MB. These files are ranked in descending order of visit times, as shown in Fig. 3. Clearly, the user access pattern of all files well matches the Zipf distribution, which conforms to the traditional point of view [3], [4].

Meanwhile, considering many literatures have reported that media files (especially videos) exhibit a special access pattern, we extract the access distribution of .rmvb files (around 70 thousand files with over 2 million visits in total) in Fig. 4 and Fig. 5, since media files are mostly in .rmvb format in QQXuanfeng. Differently, the user access pattern of .rmvb files deviates greatly from the Zipf distribution (see Fig. 4), but approximates the SE (stretched exponential) distribution proposed in [5] (see Fig. 5).

There exist two widely accepted points of view about the reason for the above non-Zipf access pattern: 1) *Fetch-at-most-once effect*. Based on their 200-day trace of KaZaa traffic, Gummadi et al. [6] declared that P2P objects are generally immutable and have a large size, so a user fetches a P2P object at most once, which is very different from the web object access pattern, namely, a user often fetches a web object (mostly a web page) repeatedly. Therefore, P2P users' access pattern deviates from web objects' access pattern (i.e., Zipf). 2) *SE in itself*. Guo et al. [5] refuted the idea in [6] by emphasizing that the media access pattern follows SE distribution in itself, rather than a deviation from Zipf, based on comprehensive analysis of 16 traces in various application scenarios and several common mathematical fitting models.

4 Understanding User Experience

Before we investigate *Problem 3: differentiated acceleration of peer swarms* (Section 4.3 in the main file), we need a practical understanding of the user experience as necessary preliminaries. Through measurements of one million XFPP (the Xuanfeng P2P protocol) peer swarms

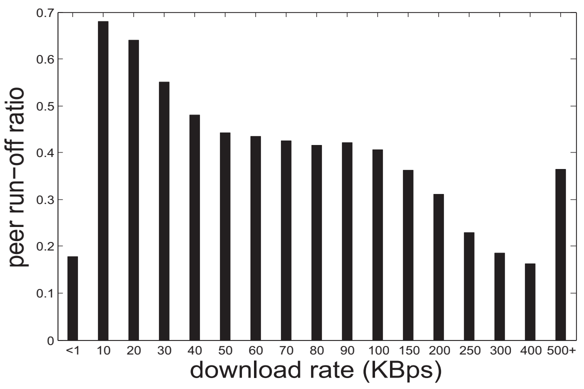


Fig. 6. Peer run-off ratio distribution over different download rate regions. “50” denotes the region [40KBps, 50KBps) and “500+” denotes the region [400KBps, $+\infty$). Note that the X-axis is not linear.

monitored by QQXuanfeng, we first analyze the peer run-off and join patterns, and then establish a simplified data supply-demand model to coarsely formalize the user experience.

4.1 Peer Run-off Pattern

Except for natural leave after finishing downloading, why does a peer *run off* (or says abnormally leave) from its affiliated swarm? This is a question people often ask but few literatures have given a satisfactory (quantitative) answer to. The reasons may include network disconnect, operating system crash, hardware failure, etc. According to our long-term operation experiences, the most common reason (accounting to nearly 70%) is that the download rate falls below the peer’s *basic expectation*; then the key point is about the concrete value of the peer’s *basic expectation*. To get this value, we measure all peers’ run-off events and plot the results in Fig. 6. The first (download rate) region possesses a very small peer run-off ratio, because most peers in this region are *seed peers* [1] which are relatively stable. The 2nd, 3rd, and 4th regions bear the largest run-off ratios, and then the run-off ratio stabilizes between 40% and 50% from the 5th region to the 11th region. Thereby, we conclude that a download rate up to 30 KBps can most efficiently prevent peers from running off. So as a rule of thumb, a peer’s basic expectation of his download rate should be:

$$d_{basic} = 30 \text{ KBps}. \quad (1)$$

Besides, we take into account a special kind of *high-demand* peers who join the swarm for live/on-demand video streaming. Observing that the basic playback rate of online videos lies between 300 Kbps and 700 Kbps, a download rate over 100 KBps (= 800 Kbps) is usually

1. As to a peer swarm, a *seed* only uploads data to *leecher*s. When the seed resides in the peer swarm, it is a *seed peer*; otherwise the seed is a *seed server* outside the peer swarm. A *leecher* downloads data from seeds and other leechers, and meanwhile uploads data to other leechers.

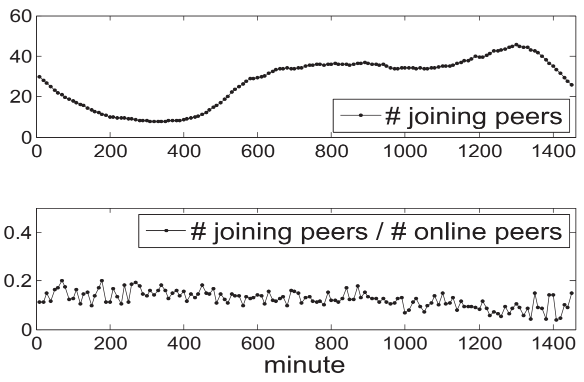


Fig. 7. Number and ratio of joining peers of a typical swarm in 24 hours.

taken as high enough to prevent the video streaming peers from running off, so we have:

$$d_{high} = 100 \text{ KBps}. \quad (2)$$

4.2 Peer Join Pattern

As to the peer join pattern, previous literatures have put forward two models in general: (1) Poisson process, i.e., peers enter a swarm at a stable rate v, but the value of v varies in different swarms [6]. (2) Zipf process [3] (or says power-law process [4]), i.e., peers enter a swarm at a dynamic rate $v(N)$ proportional to the current swarm scale N (“swarm scale” is the number of online peers inside a peer swarm.), i.e., $v(N) = \alpha \cdot N$. According to our measurements in QQXuanfeng, the Zipf process is more suitable for modeling the peer join pattern. For example, Fig. 7 plots the number and ratio of joining peers of a typical swarm in 24 hours, starting from 0:00, GTM+8. It illustrates that the proportional factor α ($= \frac{\#\ joining\ peers}{\#\ online\ peers}$) is basically stable around 0.13. On the contrary, the number of joining peers varies greatly in different hours, so the Poisson process is unsuitable. Besides, since peers are mainly composed of leechers, the Zipf process is also applicable to the leecher join pattern.

4.3 User Experience: A Simplified Supply-Demand Model

Here we use a simplified supply-demand model concerning peers’ download, upload, run-off and join behaviors, to *coarsely understand* the user experience inside a peer swarm. Given a peer swarm in a stable state at time t, whereas “stable state” means the swarm scale is stable, without abnormal run-off or join events. Suppose the current number of seeds (including both seed servers and seed peers) is S, the current number of leechers is L, the average upload rate of all nodes (including both seeds and leechers) is U, and the average download rate of all leechers is D. The peer run-off ratio $r(x)$ varies with the download rate x according to Fig. 6. Since the total upload rate must be equal to the total download rate in the swarm, we have: $(L+S)\cdot U = L\cdot D$, and then $D = \frac{L+S}{L}\cdot U$.

From time t to t' ($t' > t$), a batch of leechers join in the swarm to download data and thus breaks the stable state. Then the swarm will experience abnormal run-off events of its leechers to regain its stable state. Suppose the number of joining leechers is αL (in the previous text we have known α can be taken as a constant). Because the number of leechers has increased from L to $L(1+\alpha)$ and the joining leechers usually cannot supply data, the only way for the joining leechers to download data is to grab from existing leechers. Thus, the average download rate of all leechers decreases to:

$$D' = \frac{L+S}{L}\cdot\frac{U}{1+\alpha}. \tag{3}$$

Obviously, $D' < D$. The reduction in download rate would typically lead to a rise in the peer run-off ratio. The increase of the peer run-off ratio is: $\delta = r(D') - r(D) > 0$, and then the increased number of running-off peers is:

$$\Delta = L(1+\alpha)\cdot r(D') - L\cdot r(D) > 0.$$

As a result, the growth in the swarm scale is:

$$\alpha L - \Delta = \alpha L + L\cdot r(\frac{L+S}{L}U) - L(1+\alpha)\cdot r(\frac{L+S}{L(1+\alpha)}U)$$
$$= L\cdot(\alpha + r(\frac{L+S}{L}U) - (1+\alpha)\cdot r(\frac{L+S}{L(1+\alpha)}U)),$$

and then the growth ratio is:

$$\frac{\alpha L-\Delta}{L} = \alpha + r(\frac{L+S}{L}U) - (1+\alpha)\cdot r(\frac{L+S}{L(1+\alpha)}U) \tag{4}$$

Our goal is to make the growth ratio $\frac{\alpha L-\Delta}{L}$ and the average download rate D' be as large as possible. Now that we can hardly alter the value of α or U because they depend on the peer activity pattern and the Internet transmission capacity, and we cannot adjust the peer run-off ratio function $r(x)$ because it rests with the peer's basic expectation, then the natural method is to increase the common variable factor in Equation (3) and (4): $\frac{L+S}{L}$. We name $\frac{L+S}{L}$ as *ATD* (availability-to-demand), where the numerator $L+S$ denotes the number of data suppliers and the denominator L denotes the number of data consumers. *ATD* represents how many suppliers are allocated to a consumer in average, i.e., the statistically data supply-demand condition inside a peer swarm. A peer swarm with low *ATD* may suffer from data supply-demand imbalance and the consequent abnormal peer run-off behavior. In order to increase ATD, we need to direct the peer swarm to obtain extra server bandwidth, which is a key role of QQXuanfeng.

5 Additional Design Details

5.1 File Link Filtering

When a file link is put into the Content Index DB, it is first processed by using the following five kinds of link filtering methods:

- *Web site filtering*: examines whether the link comes from an "appropriate" web site. Here "appropriate" is manually defined, e.g., many sex, violence and politics relevant web sites are regarded as not appropriate.
- *Keywords filtering*: examines whether the link contains "illegal" keywords. Similarly, "illegal" is also manually defined.
- *Crawling-depth filtering*: examines whether the link is beyond the assigned *crawling depth* (note that any BFS traversing on the Internet must have a *crawling depth* to restrict its traversing coverage). Currently, the crawling depth is set as 3 link hops.
- *URL-depth filtering*: examines whether the link is beyond the assigned *URL depth*. For example, the file link "http://dl_dir.qq.com/invc/cyclone/ QQdownload2.exe" has a *URL depth* of 4 because it contains 3 "/"s. The maximum URL depth is set as 10 at this moment.
- *De-duplicate filtering*: examines whether the link is duplicated in the Content Index DB by utilizing the *bloom filter* data structure [7]. The bloom filter we use is an array of 128 bits (i.e., $m = 128$) with $k = 4$ independent hash functions. Suppose the Content Index DB contains n links in total, the false positive rate of the bloom filter should be:

$$f \approx (1 - e^{-\frac{k\cdot n}{m}})^k = (1 - e^{-\frac{n}{32}})^4. \tag{5}$$

Since n is very large, f can be taken as zero.

5.2 Adapting to Swarm Dynamics

Because a peer swarm is unstable with constantly changing members, the data scheduling strategy should have some basic prediction of the swarm dynamics. In other words, we should get prepared before things get worse. For example, if we allocate extra server bandwidth to a swarm only when the swarm is hungry, it would be late to prevent peers from running off. Instead, we should prepare extra server bandwidth for hungry swarms in advance, and then make necessary adjustments according to the actual situations. Based on comprehensive observations, we discover a simple prediction method which can reasonably predict the swarm status with acceptable precision.

First, we measure the *leecher ratio* distribution of all the swarms in each hour in one day, as depicted in Fig. 8. The leecher ratio denotes the ratio of the number of active leechers in an hour over the total number of active leechers in one day. And nine typical swarms are extracted to compare with Fig. 8, as plotted in Fig. 9. These swarms are sorted in descending order of leecher scale. From the above two figures we find that as a whole, the leecher ratio distribution of all the swarms exhibits an obvious diurnal pattern. And as to the leecher ratio distribution of an individual swarm, generally speaking, the larger the leecher scale is, the more similar to the curve in Fig. 8 the distribution will

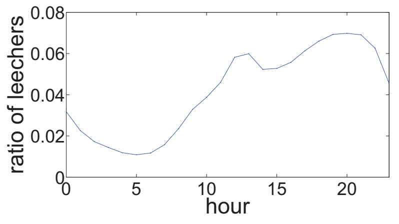


Fig. 8. Leecher ratio distribution of all the swarms in each hour in one day.

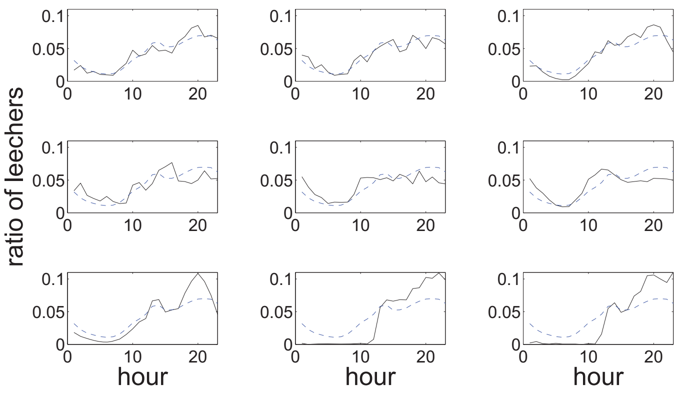


Fig. 9. Leecher ratio distributions of nine typical swarms. The blue and dotted curve corresponds to the curve in Fig. 8.

be. As a result, once the leecher scale in one day has been known or predicted, we can approximately deduce the leecher scale in each hour, so that our data scheduling strategy can proactively predict the basic status of each swarm in each hour.

Second, we measure the day-by-day status evolution of all the swarms in the year 2011. Measurement results illustrate that the status of a swarm (including the swarm scale, seed peer scale, and leecher scale in each hour) generally varies smoothly in two consecutive days, with rare *abrupt* (i.e., more than 20%) additions or reductions. Therefore, for each swarm, we can simply use its status yesterday to predict its status today. To get more accurate prediction, we use the past seven days' data (i.e., their mean value) to predict the status of a swarm today. Using the historical data of more than seven days cannot improve (but may even reduce) the prediction precision.

Specifically, as shown in Table 3, the relative error of prediction ($= |\frac{predicted\ scale - meansured\ scale}{meansured\ scale}|$) decreases as the swarm scale increases. For a large swarm (i.e., swarm scale > 1000), the relative error of prediction is usually below 10%; for a small swarm (i.e., swarm scale < 10), the absolute error of prediction is usually within 10. In conclusion, our proposed method can reasonably predict the swarm status with acceptable precision.

5.3 ISP-friendly Cache

In recent years, a variety of ISP-friendly mechanisms (e.g., [8], [9], [10]) are proposed and implemented to alleviate the tension between P2P systems and ISPs.

TABLE 3
Relative error of our prediction method.

Swarm s-cale	Relative error of swarm scale prediction	Relative error of seed peer scale prediction	Relative error of leecher scale prediction
> 10000	4.35%	7.38%	9.92%
> 1000	7.35%	7.46%	19.39%
> 100	12.51%	12.16%	31.05%
> 10	17.29%	16.72%	54%

Our QQXuanfeng system has also implemented its ISP-friendly cache mechanism in some cooperative ISPs in China, in order to reduce the cross-ISP network traffic and meanwhile improve its users' download rates.

Every day, the index information of the latest N most popular files is pushed from the QQXuanfeng Content Index DB to each cooperative ISP. N is configured as 10,000 at the moment. Each cooperative ISP only needs to provide a *cache server* and an *ISP tracker*. In fact, on one extreme, the cache server and ISP tracker can run on top of a single machine; on the other extreme, the cache server can be composed of multiple physical machines. For each cooperative ISP, its cache server stores as many popular files as possible (according to the storage capacity) and its ISP tracker maintains the index information of the cached files. Because the index information of popular files is updated per day, the cache server should update its stored files accordingly. If a novel popular file has not been stored in the cache server, the cache server needs to download the file as a common QQXuanfeng client does.

In a cooperative ISP, when a client wants to download a file f, it first sends the request to the corresponding ISP tracker, which further tells the client whether and where f can be downloaded *within this ISP*. If f can be downloaded within this ISP, the client first attempts to retrieve f within this ISP but not from the cache server. Furthermore, if its download rate falls below 30 KBps, the client is allowed to retrieve additional data from the cache server. On the other hand, if f cannot be downloaded within this ISP, the client is redirected (by the ISP tracker) to the QQXuanfeng Content Index DB.

Our ISP-friendly cache mechanism imposes extra traffic and expense on cooperative ISPs, such as extra server machines, inter-ISP download traffic of the cache server, monitoring traffic of the ISP tracker, and so forth. But its benefit is remarkable, usually far exceeding the extra cost. Since our ISP-friendly cache mechanism has still been in its startup stage, it looks simple and straightforward so there remains considerable optimization space.

6 Conclusion and Future Work

In this paper, we first review today's three major Internet content distribution modes: (a) *server-based*, (b) *P2P* and (c) *P2SP*. Although P2SP can provide efficient hybrid server-P2P content distribution, it generally works in a closed manner by only utilizing its private owned servers to accelerate its private organized peer swarms. Consequently, P2SP still has its limitations in both content abundance and server bandwidth. To this end, the fourth mode (or says a generalized mode of P2SP) has appeared as (d) *open-P2SP* which integrates various third-party servers, contents and data transfer protocols all over the Internet into a large, open and federated P2SP platform. Based on a large-scale commercial open-P2SP system named QQXuanfeng, this paper investigates the key challenging problems, practical designs and real-world performances of open-P2SP.

An important future work is about the setting of system parameters. This paper reveals several valuable system parameters via comprehensive measurements, such as the users' basic expectation of download rate: $d_{basic} = 30$ KBps, the limitation of extra server bandwidth utilization: $EBU = 40\%$ and so forth. Although the current QQXuanfeng system generally works well with these parameters, it is hard to say these parameters are the best and can well adapt to possible significant changes of underlying network environments or user requirements. It will be interesting and useful to explore how to design some mechanisms to automatically collect and analyze measurements got from servers and peers, and thus dynamically tune the system parameters to match new situations.

Besides, recently there has been a potential trend in integrating open-P2SP service into web browsers to transparently accelerate the common web (HTTP/FTP/RTSP) download. To our knowledge, at least three popular web browsers [11], [12], [13] have (partially) implemented this service. Undoubtedly, the combination of open-P2SP and web browsers will greatly benefit web users, and their developers may obtain useful heuristics from the designs of QQXuanfeng.

7 Acknowledgements

This work is supported in part by the China 973 Grant. 2011CB302305, China NSF Grant. 61073015, and US NSF Grants. CNS-0905037, CNS-1017092, CNS-1017647 and CNS-1117536. We would like to thank the reviewers and editors of our manuscript for their detailed and valuable comments, and Miss Adele Lu Jia from Delft University of Technology for her helpful advice.

References

[1] BitTorrent web site. http://www.bittorrent.com.

[2] eMule web site. http://www.emule.org.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. "Web caching and Zipf-like distributions: Evidence and implications," Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Mar. 21 – 25, 1999, New York, NY, US, pp. 126 – 134.

[4] M. Faloutsos, P. Faloutsos, and C. Faloutsos. "On power-law relationships of the internet topology," Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM), Aug. 31 – Sep. 3, 1999, Cambridge, MA, US, pp. 251 – 262.

[5] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "The stretched exponential distribution of internet media access patterns," Proceedings of the 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), Aug. 18 – 21, 2008, Toronto, Canada, pp. 283 - 294.

[6] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), Oct. 19 – 22, 2003, Sagamore, NY, US.

[7] A. Broder, and M. Mitzenmacher. "Network applications of bloom filters: A survey," Internet Mathematics, vol. 1, no. 4, pp. 485 – 509, 2004, published by AK Peters.

[8] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. "P4P: Provider portal for applications," Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM), Aug. 17 – 22, 2008, Seattle, WA, US.

[9] D. Choffnes, and F. Bustamante. "Taming the torrent," Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM), Aug. 17 – 22, 2008, Seattle, WA, US..

[10] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang. "TopBT: a topology-aware and infrastructure-independent BitTorrent client," Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), March 15 – 19, 2010, San Diego, CA, US.

[11] TT web browser. http://tt.qq.com.

[12] Sogou web browser. http://ie.sogou.com.

[13] 360 web browser. http://se.360.cn.