2010 First International Conference on Networking and Distributed Computing

Traffic Locality in the eMule System

Lijie Sheng, Xuewen Dong, Jianfeng Song, Kun Xie
School of Computer Science & Technology, Xidian University, Xi'an 710071, China
ljsheng@xidian.edu.cn

***Abstract*—Traffic locality is a promising way for P2P systems to reduce cross-network traffic and improve performance. For the specific application, eMule, there are few reports on traffic locality in eMule to date. In this paper, we study how to integrate traffic locality mechanism into the eMule system. The eMule protocol is summarized and the source code of the eMule software is analyzed. Then traffic locality mechanisms for eMule are proposed. The methods are based on measurement with ICMP ping packet to get hop distance and RTT between peers. Then a peer chooses peers with shorter hop distance or RTT as sources to connect to and to request for downloading. Simulation experiments show that such simple mechanisms can cut-down cross-network traffic greatly and ameliorate download performance of the eMule system explicitly at the same time.**

Keywords- P2P; eMule; traffic locality

I. INTRODUCTION

Peer to Peer (P2P) software plays a significant role on Internet stage. Various P2P software, such as file downloading [1,2], media streaming [3], and Voice over IP [4], brings pleasure and convenience to users. However, Internet Service Providers (ISPs) do not like P2P, because their networks are flooded by P2P traffic. According to statistics of IPOQUE, P2P software generates a part of 50% to 70% traffic on some backbone networks [5]. Such huge traffic demonstrates the popularity of P2P software. On the other hand, it discloses the issue which P2P software must face for evolution, i.e. the ISP friendliness problem: when P2P software constructs overlay network, they neglect the underlay network topology, and choose neighbors randomly, so most of the connections are between far away peers, and most of the traffic crosses several networks. Measurement of BitTorrent shows that when there are data copies in local network, parts of 70% to 90% are still downloaded from external networks [6]. Such ignorance of network topology and choosing not near peers but remote ones as data sources, result in great amount of cross-network traffic which occupies valuable inter-network links and increases financial burdens on ISP. For P2P users, since the inter-network links are usually bottlenecks of the networks and are more congested than other links, choosing remote sources leads to large Round Trip Time (RTT) and slow transmission rates, i.e. bad user experience.

One way to solve the ISP friendliness problem is P2P cache [7-11], i.e., placing P2P cache in ISP networks to cache P2P traffic. It is found that files transmitted in P2P system are usually downloaded by many peers, so it can be cached [7]. The popularity and file size distribution are measured [9]. The file popularity in P2P system is not like that of Web objects to be Zipf-like distribution, but can be modeled by a Mandelbrot-Zipf distribution. The file size distribution can be modeled by several Beta distributions, i.e., files transmitted in P2P networks are larger than web objects. Those measurement results infer that it is necessary to design dedicated P2P cache algorithms rather than using classical cache algorithms or Web cache algorithms. Several P2P cache algorithms are proposed in [8,9], and they can achieve a byte hit rate up to 35%. Another solution cuts P2P object into small pieces and encapsulate them in HTTP protocol, and then using HTTP cache realizes P2P cache [10]. A realistic P2P cache system is implemented in [11], and it is found that P2P cache is more complicated than other caches, and specialized storage system and cache algorithms are needed.

The problem of P2P cache is that it needs to communicate with various active P2P software and to speak their protocols, so the workload is high. Moreover, P2P objects are usually with big size [5,9], so it needs huge-size and high-speed cache to accommodate those P2P objects. The cache may become a single point of failure and bottleneck.

Another way is the localized download strategy, i.e., peer prefers to choose nearby peers to construct overlay network, and to download from close sources. The idea is firstly proposed in [12], which suggests choosing peers in the same ISP as neighbors to connect to. Simulation experiments demonstrate that the so-called "biased neighbor selection" can decrease cross-network traffic greatly and keep good download performance at the same time. However, the paper does not mention how to judge whether two peers are in the same ISP.

Network coordinate systems can be used to estimate the distance between peers. Network coordinate systems map network nodes to low dimension Euclidean space by some end to end measurements based on the assumption that the distance (represented by delay) between network nodes usually corresponds with triangle inequality. After the mapping, for any two network nodes, the distance between them can be calculated by their coordinates. A typical network coordinate system, Vivaldi, is chosen to use in Azureus (now Vuze) [13], because of its high resolution and good scalability. Field tests show that with some optimization the measurement resolution is good. However, in most network coordinate systems, there are circumstances that violate the triangle inequality, which affects the estimate resolution. Moreover, network coordinate systems describe distance with delay, but delay is not the only factor which affects P2P performance, and loss rate and available bandwidth also affect download performance.

Ono proposes to exploit CDN as a reference system to determine whether two peers are close [14]. When two peers get similar server list from dynamic DNS system of the CDN,

978-0-7695-4207-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ICNDC.2010.80

it implies they are near to each other, and should have higher priority to set up a connection between them. Ono is developed as a plug-in in Azureus (now Vuze) BitTorrent client. Statistics on hundreds of thousands of users show that Ono can decrease cross-network traffic effectively. Especially, over 33% of the time it selects peers in the same AS. Loss rate decreases by 30% and the average download rate increases by 31%. The problem of the method is that it relies completely on the CND system, and the nearby information is coarse.

P4P suggests that the topology information should be provided by ISP [15], because they have the most detailed information about the network they operate. In P4P architecture, a server named iTracker provides information such as network topology and traffic engineering strategy, and is queries by P2P software. Simulation and field test discover that P4P can decrease download time by 20%, and can decrease cross-network traffic by 50% to 70%. The advantage of P4P is that ISP can provide not only the nearby information, but also traffic engineering strategy to guide P2P traffic to low load and low cost links, which solves the Application Layer Traffic Optimization (ALTO) problem [16-18]. However, P4P needs cooperation and trust between ISP and P2P software, and a protocol for communication between them must be designed and accepted, which takes time.

A similar solution proposes that ISP provide an oracle to help peer choose neighbors [19]. The peer sends to the oracle a peer list. Then the oracle sorts the peers according to some localized strategy and returns back the list to the peer. Then the peer connects to some peers as neighbors according to the list. The policy according to which oracle sorts peers is flexible. For example, it can prefer peers in the same ISP, or peers nearby in geography, or peers with which there are high throughput. Simulation and field tests report that the solution is effective.

To date most of the researches and experiments on traffic locality are carried on BitTorrent [12,13,14,15,20,21], and other P2P applications are seldom concerned [22,23]. Especially, there are few reports on traffic locality with the eMule software [23]. This is unreasonable because eMule is a very popular software with millions of daily users [5]. On the other hand, eMule is different from BitTorrent, especially in their credit and incentive systems. Both of them use tit-for-tat strategy to encourage uploading, but the implementation of the strategy is dissimilar. In eMule peer records upload and download data amount of other peers and calculates the ratio between them as credit values, and synthesizes credit value, waiting time and file priority into a score, and then queues peers according to their scores. Peers with high scores will be chosen to upload to. While BitTorrent uses choke/unchoke mechanism and usually unchoke peers with highest uploading rates to the peer. Different mechanisms lead to different overall behaviors. Thus it is necessary to study how to localize traffic in the eMule system, which is the main concern of this paper. Our finding is that modifying the requesting phase of the eMule system and making a peer prefer to request for downloading from nearby sources can decrease cross-network traffic effectively and can increase download performance at the same time. Comparing with the scheme in [23], the method in this paper does not have explicit drawback, and is more suitable to be used in the real-world eMule system.

The remainder of the paper is organized as follows. Section II summarizes the eMule system. Section III proposes the traffic locality methods in the eMule system. Section IV describes the evaluation methodology and criteria, and then shows the experiment results. Section V concludes the paper.

II. THE EMULE SYSTEM

The eMule protocol is based on eDonkey protocol, and the former extends the communication contents of the latter. The eMule network is composed of hundreds of eMule servers and tens of millions of eMule clients. To obtain service from server, an eMule client must connect to an eMule server. During the downloading process of a file, the eMule client must connect to other clients, and join in their waiting queue. When the score of the client is high enough to reach the top of waiting queue, the client can begin downloading.

For client-server communication, the client publishes shared file list to server, and gets source list for downloading file from server periodically.

EMule can download many files simultaneously, and each file can be downloaded from many sources at the same time. After getting source list from server, an eMule client will connect to sources in the list. If the source has data blocks that the client needs, the client will request for uploading from the source. If the size of the waiting list of the source does not exceed a limit, the source will accept the request and put the client in its waiting list.

At the source end a client will measure the upload bandwidth periodically to judge whether it can upload to new client. Credit values and scores of the clients are calculated, and the client with a higher score in waiting list will preempt the client in uploading list. Credit value records upload and download data amount of a client. The score value is computed according to three factors: waiting time, credit value, and priority of the file. When there is enough bandwidth or a client in waiting list has higher score, the source client will upload to the new client. When a source accepts a client's uploading request, the client will send block requests to the source. EMule prefers to choose the following blocks: nearly completed, useful for preview, or scarce in the eMule network. When the source receives the block requests, it will process the requests and send data blocks to the client.

III. TRAFFIC LOCALITY METHODS IN EMULE

In standard eMule implementation, when a client gets source list from server or Kad network, it selects N sources in the list randomly to connect to and request for uploading. We modify the mechanism in following way: when the client gets source list, it selects N nearest sources to connect to; if waiting list of a source is full and the source can not accept the request from the peer, the peer eliminates the source from list and tries the next nearest source. Thus, in the new eMule system, every peer still connects to N sources, and the N sources are the nearest and available N sources.

The distance between peers can be obtained in several ways. For example, we can use CDN as a reference system to determine whether two peers are close, just like [14]. The P4P

architecture [15] can also be used, i.e., we utilize topology information provided by ISP to calculate the distance between peers, e.g. how many hops between peers. In another way, peers measure distance between each other by themselves. In this paper, we focus on the last way.

The modified eMule scheme works like this: when a peer get source list, it measures the distance to those sources, and then chooses N nearest and available sources to connect to. The distance between peers can be measured by sending and responding packets. If a peer is not sure whether the source is a modified version just like itself, it sends ICMP ping packet to the source. ICMP ping packet is actually an ICMP Echo-Request packet. When the IP protocol stack of the source peer receives the ICMP Echo-Request packet, it will reply an ICMP Echo-Reply packet to the client. Two kinds of information can be obtained as distance: one is TTL, and the other is RTT.

Time To Live (TTL) is a field in IP packet header. It is filled with an initial value at the sender. Each time the packet passes a router, its TTL value is decreased by one. When the receiver receives the packet, its TTL value can be compared with the initial value, and their difference is the number of routers the packet passed, which is called hop distance in this paper. Depending on the operating system of remote peers, the initial value of TTL can be 255 (FreeBSD, Solaris), 128 (Windows NT4/2000/XP), 64 (Linux kernel 2.2, 2.4, 2.6), and 32 (Windows 95/98/ME). Since most of Internet paths have hops of no more than 30 [24], when the peer receives the responding packet from the source, it can infer the initial value of TTL, and then get the hop distance to the source. When hop distance is used as the distance between peers, the modified eMule scheme is denoted by "TTL eMule".

Another metric is RTT (Round Trip Time). The peer sends out an ICMP ping packet and receives the responding one. Then the time between the sending and receiving actions of the peer is RTT. To handle with the fluctuation, several ICMP ping packets can be sent and the average RTT can be computed. Moreover, like the RTT measurement in TCP [25], a weighted average RTT can be calculated as $SRTT = (1-\alpha) \times SRTT + \alpha \times RTT$, in which SRTT stands for smoothed RTT. α can be set as 1/8 according to [25]. When RTT is used as the distance between peers, the modified eMule scheme is denoted by "RTT eMule".

If both the peer and the source peer implement the modified eMule, then they can exchange packets in application level to measure hop distance or RTT. IP raw socket can be used, and the initial value of TTL can be set freely.

IV. Experiments and Evaluation

A. Methodology

To evaluate the traffic locality method in eMule accurately, an eMule simulator was developed according to the source code of the eMule software. The simulator is a discrete event simulator, developed with standard C++ and STL. Details of the simulator can be found in [26].

The simulated network is a Transit-Stub network generated by GT-ITM [27]. There are one Transit network and 12 Stub networks. Each Stub network contains 15 Stub routers. Each Stub router connects to several end nodes. Each end node represents an eMule client. In all experiments, all clients are distributed symmetrically in the 12 Stub networks. The upload and download bandwidth of end node is 10Mbps. Bandwidth of all the other links is 1Gbps. The propagation delay of links between Stub nodes and end nodes is 1 millisecond. The propagation delay of links between Stub nodes is uniformly distributed between 1 and 5 milliseconds. The propagation delay of links between Transit nodes and Stub nodes is uniformly distributed between 5 and 10 milliseconds. The propagation delay between Transit nodes is uniformly distributed between 10 and 20 milliseconds.

B. Metrics

The following evaluation criteria are used in this paper:

- Download time: download time of all clients are recorded, and are illustrated by cumulative distribution function (CDF).
- Bandwidth utilization of cross-network links: in the network topology described above, the cross-network links are between Transit nodes and the corresponding Stub networks. Average bandwidth utilization of all these links is calculated.
- ISP traffic redundancy: amount of data going into each Stub networks is recorded. Then divide the average data amount by shared file size, we get ISP traffic redundancy [12], which means how many times each block of the shared file goes into the Stub network when all peers finish downloading.
- Unit Bandwidth Distance Product (unit BDP): multiply the transfer data amount by the number of links they travel, we get bandwidth distance product. For each client, add all BDP for data it receives, then divide it by the file size, we get unit BDP [15], which means averagely how many links each block of data travels to the client. CDF is used to show the distribution of unit BDP for all clients.

C. Results

The experiments are conducted on a PC with Intel Pentium D 2.8GHz processor and 1GB RAM running Ubuntu 8.04 desktop version. The regular eMule (denoted by "Regular"), RTT eMule (denoted by "RTT"), and TTL eMule (denoted by "TTL") are compared. In all experiments, the shared file size is 1.172GB; the file is divided into 120 parts; each part is composed of 53 blocks; the block size is 184320 bytes.

Figure 1, Figure 2, Figure 3, and Table I compare the three eMule schemes in download time, bandwidth utilization of cross-network link, unit BDP, and ISP traffic redundancy when the number of peers in system is 2160. The results show that the effect of "RTT" is similar with that of "TTL", and is much better than that of "Regular". "RTT" and "TTL" reduce average download time by 26%, decrease average unit BDP by 42%, and cut down ISP traffic redundancy by 66%, compared with "Regular".

TABLE I. PERFORMANCE FOR 2160 PEERS WITH THE THREE EMULE SCHEMES

The eMule scheme	Average download time	ISP traffic redundancy	Average unit BDP
Regular	2311.87	153.62	11.29
RTT	1701.21	47.95	6.53
TTL	1697.57	52.11	6.39

Next we vary the number of peers in system to see the benefits of traffic locality. Figure 4, Figure 5, and Figure 6 plot average download time, ISP traffic redundancy, and average unit BDP with the three eMule schemes when the number of peers in system varies from 1080 to 4320. In all experiments, peers are distributed in 12 Stub networks symmetrically. Figure 4 and Figure 5 show that average download time and ISP traffic redundancy with "RTT" and "TTL" hold on horizontal lines approximately when the number of peers in system increases, while those with "Regular" grows linearly. It means traffic locality can decouple download time and cross-network traffic from the number of peers in system, while regular eMule can not. Figure 6 shows that average unit BDP of the three schemes decrease slowly when the number of peers increases. The reason is that when there are more peers in system, each peer has more neighbors nearby from which they can download from.

V. CONCLUSION

In this paper we study the traffic locality methods in eMule system. Two simple mechanisms are proposed, and their effects are demonstrated by simulation experiments. The mechanisms measure hop distance and RTT between peers with ICMP ping packet, and then choose peers with small hop distance or RTT as sources to connect to and request to download from. Simulation results show that the mechanisms can decrease peers' download time and cut down cross-network traffic explicitly at the same time. It means that traffic locality can benefit both eMule users and ISP. Moreover, the mechanisms are very simple, and do not need any help from external infrastructure, so it offers a simple way for eMule to improve performance and ISP friendliness simultaneously.

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for the Central Universities of China (JY10000903012).

REFERENCES

[1] B. Cohen, "Incentives build robustness in BitTorrent," In Proc. of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon) , 2003.

[2] EMule, http://www.emule-project.net.

[3] Y. Huang, T. Z. J. Fu, D. Chiu, J. C. S. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," ACM SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

[4] H. Xie and Y. R. Yang, "A measurement-based study of the skype peer-to-peer VoIP performance," In Proc of IPTPS, Bellevue, WA, Feb. 2007.

[5] IPOQUE, "Internet Study 2008/2009: The Impact of P2P File Sharing, Voice over IP, Instant Messaging, One-Click Hosting and Media Streaming on the Internet," February 2009.

[6] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05), Berkeley, CA, USA, Oct. 2005, pp. 63–76.

[7] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, "Are file swapping networks cacheable? Characterizing P2P traffic," in Proc. of WWW Caching Workshop, Boulder, CO, USA, Aug. 2002.

[8] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache replacement policies revisited: The case of P2P traffic," in Proc. of the 4th International Workshop on Global and Peer-to-Peer Computing (GP2P'04), Chicago, IL, USA, Apr. 2004.

[9] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," The 14th IEEE International Conference on Network Protocols (ICNP06), Santa Barbara, California, November 12-15, 2006.

[10] G. Shen, Y. Wang, Y. Xiong, B. Zhao, and Z. Zhang, "HPTP: relieving the tension between ISPs and P2P," In Proc. of International Workshop on Peer-to-Peer Systems (IPTPS'07), Bellevue, WA, February 2007.

[11] M.Hefeeda, C. Hsu, and K. Mokhtarian, "Design and Evaluation of a Proxy Cache for Peer-to-Peer Traffic," Technical Report, School of Computing Science, Simon Fraser University, July 2008.

[12] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," In Proc. of the Int'l Conference on Distributed Computing Systems (ICDCS) , 2006.

[13] J. Ledlie, P. Gardner, and M. Seltzer, "Network Coordinates in the Wild," Proceedings of NSDI, Cambridge, MA, April 2007.

[14] D. Choffnes and F. E. Bustamante, "Taming the Torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems," ACM SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

[15] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

[16] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," RFC 5693, Oct. 2009.

[17] S. Kiesel, L. Popkin, S. Previdi, R. Woundy, and Y R. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements," IETF Internet-Draft, draft-ietf-alto-reqs-02.txt, Oct. 2009.

[18] V. K. Gurbani, V. Hilt, I. Rimac, M. Tomsu, and E. Marocco, "A Survey of Research on the Application-Layer Traffic Optimization Problem and the Need for Layer Cooperation," IEEE Communications, Vol. 47, No. 8, pp. 107-112, August 2009.

[19] V. Aggarwal, A. Feldmann, and C. Scheidler, "Can ISPs and P2P systems cooperate for improved performance?" In ACM SIGCOMM Computer Communications Review (CCR), 37:3, pp. 29-40, July 2007.

[20] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-Awareness in BitTorrent-Like P2P Applications," IEEE Transactions on Multimedia, Vol. 11, No. 3, April 2009.

[21] X. Weng, H. Yu, G. Shi, J. Chen, X. Wang, J. Sun, and W. Zheng, "Understanding Locality-Awareness in Peer-to-Peer Systems," International Conference on Parallel Processing - Workshops, 2008.

[22] Y. Liu, L. Guo, F. Li, and S. Chen, "A Case Study of Traffic Locality in Internet P2P Live Streaming Systems," Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), Montreal, Canada, June 22-26, 2009.

[23] L. Sheng, J. Song, X. Dong, and K. Xie, "Application Layer Traffic Optimization in the eMule System," The Fifth International Conference on Internet and Web Applications and Services, ICIW 2010, Barcelona, Spain, May 9-15, 2010.

[24] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the internet," In IEEE GLOBECOM'98, Sydney, Australia, November 1998.

[25] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, November 2000.

[26] L. Sheng, J. Song, X. Dong, and L. Zhou, "EMule Simulator: a Practical Way to Study the eMule System," The Ninth International Conference on Networks, ICN 2010, Menuires, France, April 11-16, 2010.

[27] GT-ITM, http://www.cc.gatech.edu/projects/gtitm/.

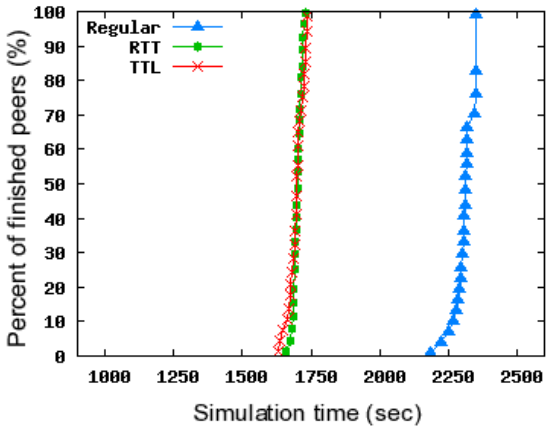

Figure 1. CDF of download time of 2160 peers with the three eMule schemes

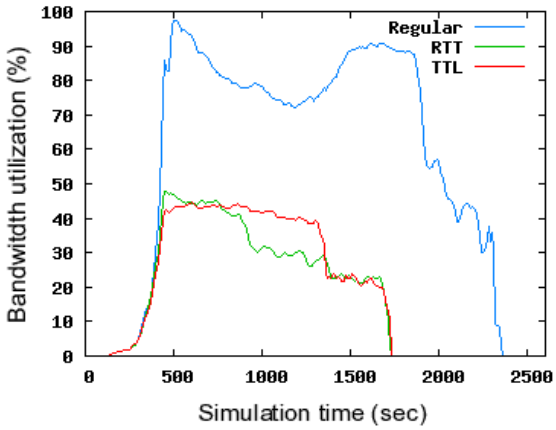

Figure 2. Average bandwidth utilization of cross-network links with the three eMule schemes

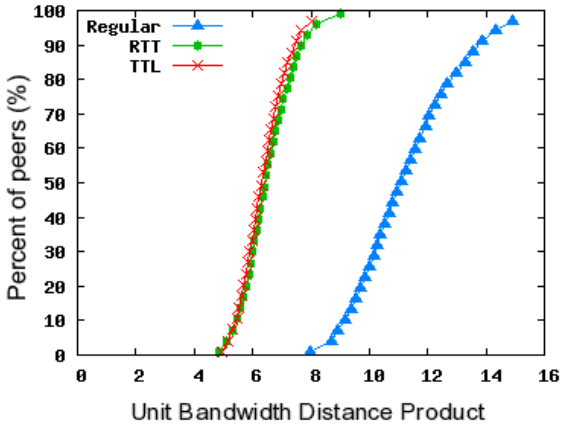

Figure 3. CDF of unit BDP of 2160 peers with the three eMule schemes

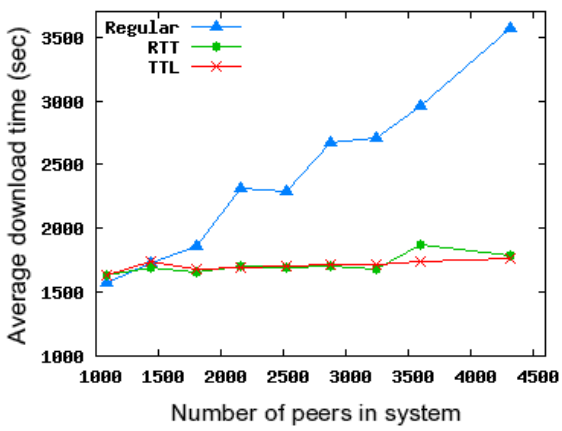

Figure 4. Average download time versus the number of peers in system with the three eMule schemes

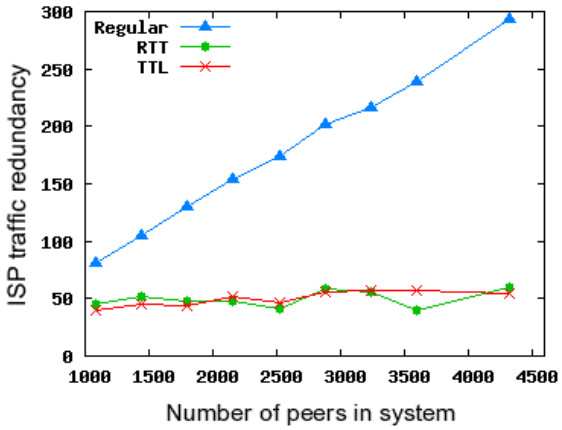

Figure 5. ISP traffic redundancy versus the number of peers in system with the three eMule schemes

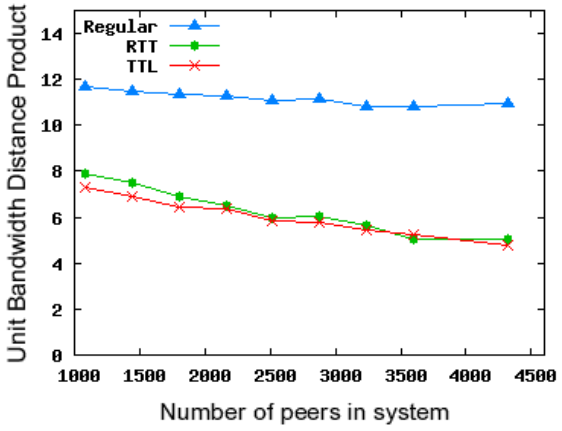

Figure 6. Unit BDP versus the number of peers in system with the three eMule schemes