International Journal of Control and Automation
Vol. 8, No. 6 (2015), pp. 89-98
http://dx.doi.org/10.14257/ijca.2015.8.6.10

A Distributed Multi-protocol Crawler based on Fuzzy Control for P2P IPTV Applications

Wenxian Wang, Xingshu Chen, Haizhou Wang* and Yi Duan

Network and Trusted Computing Institute, College of Computer Science, Sichuan University, Chengdu, China
catean@scu.edu.cn, chenxsh@scu.edu.cn, whzh.nc@qq.com, 8403428@qq.com

Abstract

With the rapid development of P2P technology, P2P IPTV applications have received more and more attention. And program-list distribution is very important to P2P IPTV applications. In order to collect IPTV program information, a distributed multi-protocol crawler was proposed based on principle of program-list distribution. The IPTV programs information will be used for characteristic analyses of program and for automatic sorting of program and establishment of IPTV repository in next work. In addition, a task scheduling model based on fuzzy control is introduced to improve performance of the crawler. In the experiment, three task scheduling algorithms are compared, and the results show that the fuzzy algorithm can balance service nodes' load effectively with less task execution time.

Keywords: *P2P IPTV, Program-list distribution, Crawler, Task scheduling, Fuzzy control*

1. Introduction

Peer-to-Peer (P2P) applications take advantage of resources such as storage, CPU cycles, content or human presence available at the edge of the Internet to provide a service [1]. With the development and maturity of P2P technology, P2P applications become more and more popular in recent ten years, which include file-sharing applications, audio-based VOIP applications, and video-based IPTV applications. However, they account for a significant proportion of Internet traffic. According to a survey from ipoque [2] in February, 2009, P2P generates most traffic in all regions. In Addition, P2P IPTV applications become popular gradually and contribute a great amount of P2P traffic to Internet [3].

Internet Protocol Television (IPTV) denotes the transport of live streams and recorded movies or video clips by means of advanced packet-switched Internet technologies [4]. In 2000, Chu proposed End System Multicast (ESM) [5], the first known P2P IPTV application, which constructs an overlay tree to distribute video data, and continuously optimizes the tree to minimize end-to-end latency. Later on, there was a proliferation of proposals that use overlay networks for efficient distribution of live video. However, they were limited in their capabilities and were not deployed in large scale. Cool Streaming was released in summer 2004 and arguably represented the first large-scale P2P video streaming experiment [6]. Since then, many P2P IPTV applications emerged in 2005. The known applications include PPTV (former PPLive), PPStream and UUSee. From 2006, related measurements of P2P IPTV were done by a number of academic staff, and we carried out the related work [7-9] from 2007.

* Corresponding author

There have been many studies about P2P IPTV measurement. The measuring methods they use can be classified in two discrete tracing approaches: passive tracing approach and active tracing approach.

The passive method is performed by deploying code at suitable points in the network infrastructure. The passive approach does not increase the traffic on the network. And it is often used to analyze and identify P2P IPTV traffic from general Internet traffic with the known behavior (*e.g.*, connection ports, feature or patterns). It is also used to capture traces analyze them to have a look about a P2P IPTV applications. Du [10] developed a machine learning methodology to identify PPLive and PPStreasm traffic. Argawal [11] studied the program startup time and the quality of service in term of number of consecutive lost block. Silverston [12] studied four IPTV applications and gave a global view of the impact of P2P media streaming on the network traffic. With abundant traces from a successful commercial P2P IPTV application, Wu [13] characterized inter-peer bandwidth availability in large-scale P2P streaming networks. The passive approach is potentially transparent, scalable and allows comparison of traffic from multiple domains side-by-side. However, it is dependent upon access to core network infrastructure, which is not always feasible. So it is often used for flow control in firewall or gateway devices.

The active method relies on special crawler, like an ordinary client, to inject test packets into P2P network or send packets to servers and peers, following them and measuring characters of P2P network. Hei [14] carried out the first active tracing of a commercial P2P IPTV application, namely, PPLive. They further developed a dedicated PPLive crawler to study the global characteristics of PPLive system [15]. Wu [16] presented Magellan to characterize topologies of peer-to-peer streaming networks of UUSee.

Most of existing research work surveyed the P2P IPTV network-centric metrics (*e.g.*, traffic characterization, TCP or UDP connections, and video traffic) or user-centric metrics (*e.g.*, user arrival and departure, geographic distribution, channel population). And none of them addressed program-list distribution. Our studies focused primarily on program-list distribution of P2P IPTV applications. A distributed program crawler was proposed to collect various kinds of information of programs. And a task scheduling model based on fuzzy control was also introduced to improve the crawler’s performance.

2. Crawling Mechanism

In this section, the basic principle of program-list distribution in P2P IPTV networks was presented, and a feasible and efficient crawler was put forward for crawling programs.

2.1. Principle of Program-list Distribution

When the program-list is downloaded and extracted by an IPTV client, one user can select a program to watch. So program-list distribution is very important to P2P IPTV applications. The program-list includes program name, categories, play-link which is the most important identification of signal communication among peers, descriptions and so on.

The client-server architecture is usually used to distribute program-list file in IPTV systems, as shown in Figure 1 [7]. When an IPTV client starts up, it requests program-list file from program-list servers and updates the local information of all programs immediately. XML is usually used in program-list files to organize various metadata of programs.

With the number of programs increasing rapidly, the size of program-list file becomes bigger and bigger. For example, PPTV had about 300 thousand programs in 2011, and the size of program-list file was more than 20MB. That is a heavy burden to program-list servers, and makes bad experience to users. Some IPTV applications use compression to decrease the file size, and others use multiple program-list files based on program categories. Furthermore, some IPTV applications encrypt program-list files to prevent hotlinking.

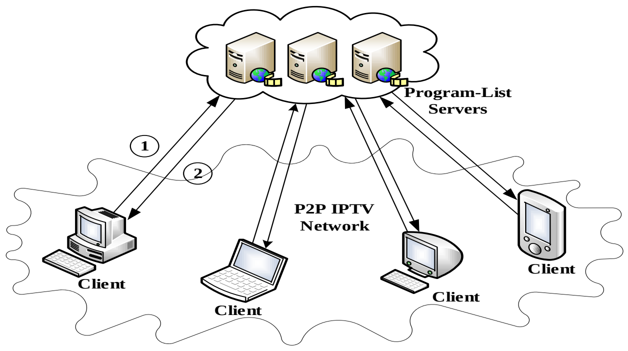


Figure 1. Program-List Distribution Architecture of IPTV

2.2. Architecture of DMP-Crawler

In order to obtain programs information of IPTV applications, two things must be done. One thing is to summarize principle of program-list distribution of most IPTV applications. The other is to decrypt the encrypt algorithm and XML metadata of program-list file.

When the two things were done, an efficient distributed multi-protocol crawler (DMP-Crawler) was proposed to collect various kinds of information of programs in popular P2P IPTV applications. {Program name, IPTV application name} was used to uniquely identify a program. Figure 2 presents an overview of architecture of DMP-Crawler, composed of one crawler controller and a number of crawler clients.

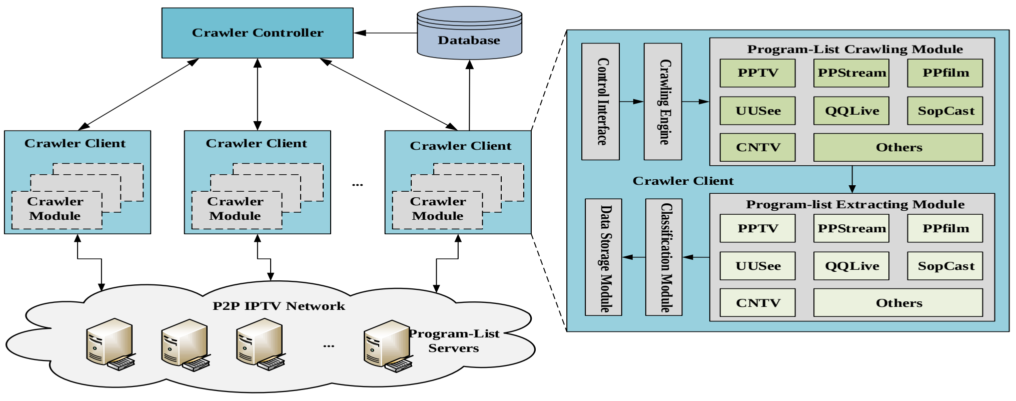


Figure 2. Architecture of DMP-Crawler

Definition 1. A crawling task is defined as the process that a crawler client collects all programs of one IPTV application at one time.

The crawler controller gets an IPTV applications list from database and initializes crawling tasks queue according priorities of all crawling tasks. Then the crawler controller pulls a task from the queue and assigns the task to an independent crawler client by a task scheduling algorithm which is on the basis of statuses of all crawler clients and servers. Each crawler client periodically reports its crawling status, CPU and memory consumption to crawler controller. When a task is allocated, the crawler client invokes crawler engine to judge IPTV application type, requests program-list file from program-list servers and reports crawling status to crawler controller. When a program-list file is downloaded, the crawler client extracts metadata of programs from the file, classifies these programs and stores all information of programs into database for further analyses.

3. Task Scheduling Model Based on Fuzzy Control

As the task scheduling and load balancing is a NP complete problem in a distributed computing environment. In order to shorten crawling time of DMP-Crawler and to balance loads of servers, fuzzy control theory is introduced to the distributed task scheduling of DMP-Crawler in this section.

3.1. Fuzzy Model of Dynamic Performance of Service Nodes

Definition 2. A service node (SN) is a computing unit used to complete tasks in the distributed computing network.

We usually call a server as a SN. In order to balance the task scheduling, it is necessary to analyze performance of all SNs and to establish a model to balance the load of SNs. CPU utilization, memory utilization, network response time, CPU reference coefficient and memory reference coefficient are chosen to evaluate the load of SNs. We define CPU utilization as U_{CPU}, memory utilization as U_{MEM}, network response time as RTT, CPU reference coefficient as C_{CPU} and memory reference coefficient as C_{MEM}. The unit of RTT is millisecond. When RTT is beyond 1000 millisecond, response of the SN is very slow and RTT is set as 1000 millisecond. C_{CPU} and C_{MEM} can be calculated by

$$\begin{cases} C_{CPU} = \dfrac{Avg(CPU)}{CPU}, CPU \geq Avg(CPU) \\ C_{CPU} = 1, CPU < Avg(CPU) \\ C_{MEM} = \dfrac{Avg(MEM)}{MEM}, MEM \geq Avg(MEM) \\ C_{MEM} = 1, MEM < Avg(MEM) \end{cases} \tag{1}$$

Where $Avg(CPU)$ is average value of CPU frequency of all SNs and $Avg(MEM)$ is average value of memory size of all SNs. Based on the above definition, the load of a SN can be expressed as

$$\begin{cases} SNLoad = W_{CPU} \times C_{CPU} \times U_{CPU} + W_{MEM} \times C_{MEM} \times U_{MEM} + W_{RTT} \times \dfrac{RTT}{1000} \\ W_{CPU} + W_{MEM} + W_{RTT} = 1 \end{cases} \tag{2}$$

Where W_{CPU}, W_{MEM} and W_{RTT} are the weights of C_{CPU}, C_{MEM} and RTT, respectively.

Definition 3. Fuzzy set of *SNLoad* is defined as ***SNL***= {Very Low, Low, Middle Low, Middle, Middle High and High}. Triangular fuzzy number [17] is used to fuzzify *SNLoad*, and the membership function (Figure 3) is defined as

$$
\begin{gathered}
\mu_{VL} = \begin{cases} 1, x \le 0.1 \\ \dfrac{0.2 - x}{0.2 - 0.1}, 0.1 < x \le 0.2 \\ 0, x > 0.2 \end{cases}
\qquad
\mu_{M} = \begin{cases} 0, x \le 0.4 \\ \dfrac{x - 0.4}{0.55 - 0.4}, 0.4 < x \le 0.55 \\ \dfrac{0.7 - x}{0.7 - 0.55}, 0.55 < x \le 0.7 \\ 0, x > 0.7 \end{cases}
\\
\mu_{L} = \begin{cases} 0, x \le 0.1 \\ \dfrac{x - 0.1}{0.25 - 0.1}, 0.1 < x \le 0.25 \\ \dfrac{0.4 - x}{0.4 - 0.25}, 0.25 < x \le 0.4 \\ 0, x > 0.4 \end{cases}
\qquad
\mu_{MH} = \begin{cases} 0, x \le 0.65 \\ \dfrac{x - 0.65}{0.75 - 0.65}, 0.65 < x \le 0.75 \\ \dfrac{0.85 - x}{0.85 - 0.75}, 0.75 < x \le 0.85 \\ 0, x > 0.85 \end{cases}
\\
\mu_{ML} = \begin{cases} 0, x \le 0.3 \\ \dfrac{x - 0.3}{0.4 - 0.3}, 0.3 < x \le 0.4 \\ \dfrac{0.5 - x}{0.5 - 0.4}, 0.4 < x \le 0.5 \\ 0, x > 0.5 \end{cases}
\qquad
\mu_{H} = \begin{cases} 0 x \le 0.8 \\ \dfrac{x - 0.8}{0.9 - 0.8}, 0.8 < x \le 0.9 \\ 1, x > 0.9 \end{cases}
\end{gathered}
\tag{3}
$$

So the fuzzy result can be expressed as

$$
f(SNLoad) = \frac{\mu_{VL}}{VL} + \frac{\mu_{L}}{L} + \frac{\mu_{ML}}{ML} + \frac{\mu_{M}}{M} + \frac{\mu_{MH}}{MH} + \frac{\mu_{H}}{H} \tag{4}
$$

For example, when *SNLoad* = 0.45, the fuzzy result can be expressed as $f(SNLoad) = \frac{0}{VL} + \frac{0}{L} + \frac{0.5}{ML} + \frac{0.33}{M} + \frac{0}{MH} + \frac{0}{H}$.

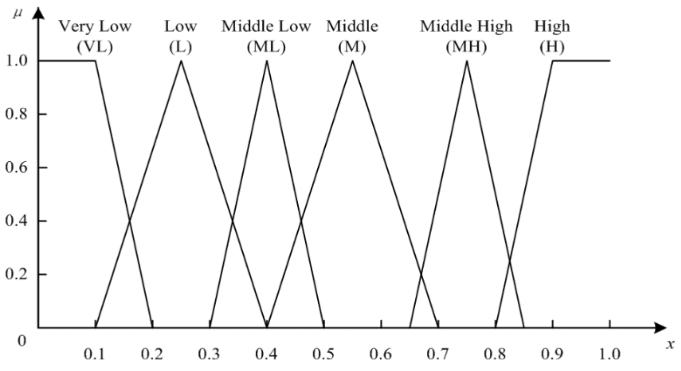


Figure 3. Membership Function

3.2. Fuzzy Model of Task Request Load

Definition 4. Task Request Load (*TRLoad*) is the load generated by a task in a SN. Fuzzy set of Task request load is defined as ***TRL*** = {Very Low, Low, Middle Low, Middle, Middle High, High}.

Task request load can be calculated by a SN's load status of whether the SN executes the task or not. To facilitate modeling of load, values of ***SNL*** and ***TRL*** fuzzy sets are translated into corresponding numbers in Table 1.

Table 1. Corresponding Number of Fuzzy Sets

Value/ Load Level	Very Low (VL)	Low (L)	Middle Low (ML)	Middle (M)	Middle High (MH)	High (H)
Number	1	2	3	4	5	6

3.3. Defuzzification

In order to obtain the load level of SNs and tasks, we adopt center of gravity method [18] to take defuzzification for *SNLoad* and *TRLoad*.

$$l(Load) = \frac{\sum x_i \times \mu(x_i)}{\sum \mu(x_i)}$$
$$= \frac{VL \times \mu_{VL} + L \times \mu_L + ML \times \mu_{ML} + M \times \mu_M + MH \times \mu_{MH} + H \times \mu_H}{\mu_{VL} + \mu_L + \mu_{ML} + \mu_M + \mu_{MH} + \mu_H} \quad (5)$$

In the front example, load level of *SNLoad* can be calculated by Equation (5), e.g.

$$l(Load) = \frac{1 \times 0 + 2 \times 0 + 3 \times 0.5 + 4 \times 0.33 + 5 \times 0 + 6 \times 0}{0 + 0 + 0.5 + 0.33 + 0 + 0} = 3.40$$

The value is closer to ML level, so load level of *SNLoad* is ML level.

3.4. Inference Rules

In the task scheduling model, the inference rules are used to select the most suitable SN to execute a task. There are six levels of task request load and six levels of service node load, and they can be designed to be related by some rules. The inference rules are expressed as follow:

If ***TRL*** value is Very Low, then ***SNL*** value is High.
If ***TRL*** value is Low, then ***SNL*** value is Middle High.
If ***TRL*** value is Middle Low, then ***SNL*** value is Middle Low.
If ***TRL*** value is Middle, then ***SNL*** value is Middle.
If ***TRL*** value is Middle High, then ***SNL*** value is Middle Low.
If ***TRL*** value is High, then ***SNL*** value is Very Low.

If the current most appropriate ***SNL*** value does not exist, select the closest ***SNL*** value. For example, if ***TRL*** value is Middle and all of ***SNL*** values are not Middle, the system will select the ***SNL*** value of Middle High or Middle Low.

3.5. Task Scheduling Process of DMP-Crawler

Based on the front task scheduling model, task scheduling process of DMP-Crawler was realized as follow:

1) Initializing scheduling queue according priority of all tasks.
2) Calculating all tasks' *TRLoad* and ***TRL*** value.
3) Computing *SNLoad* and ***SNL*** value of all SNs in a fixed period.
4) Selecting the most appropriate SN to complete the task by inference rules when the crawler controller pulls a task from the scheduling queue.

4. Results and Analysis

DMP-Crawler crawls 33 IPTV applications in China every day, and there are 33 tasks for crawler controller to assigns to crawler clients through the task scheduling model based on fuzzy control. Moreover, we can added or remove crawling tasks in the future. Therefore, task scheduling experiments were conducted in two conditions. One condition is 6 tasks scheduled on 3 servers, and the other is 100 tasks scheduled on 5 servers. ***TRL*** values of all tasks were showed in Table 2.

In the experiments, a task is the process that a testing downloads a txt file from a Web server and calculates the number of words in the file. In the tow experiments, all servers have the same configuration and were deployed in a LAN. Let W_{CPU}=0.5, W_{MEM}=0.4 and W_{RTT}=0.1.

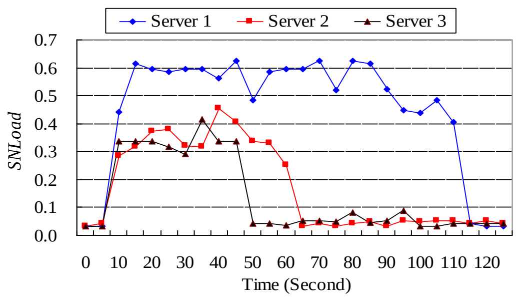
Three task scheduling algorithms are used to compare scheduling performance including load balance and execution time of tasks. The first is random scheduling algorithm (Random), which randomly selects a server to execute a task. The second is the fuzzy scheduling algorithm (Fuzzy) proposed in this paper, which selects the 'best' server to perform a task. The third is also based on the fuzzy scheduling algorithm, but it selects a server with the lightest load to execute a task, and is also called fuzzy optimal choice algorithm (Fuzzy-Best).

Table 2. TRL Values of All Tasks

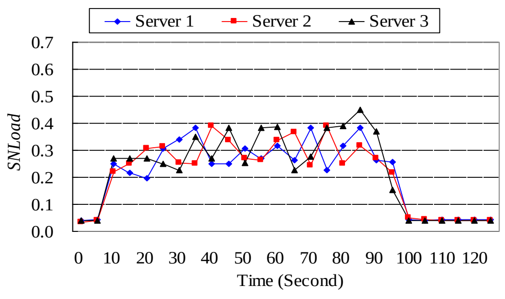
TRL value	Number of task	
	Experiment I	Experiment II
1	1	20
2	1	20
3	1	10
4	1	15
5	1	15
6	1	20

4.1. Load of SNs

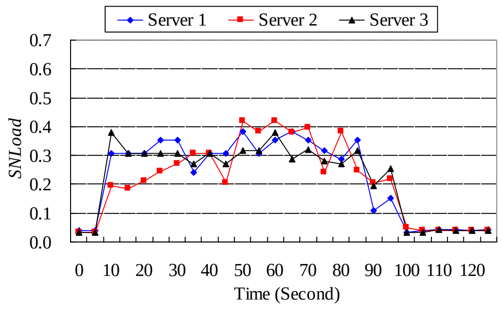
The results of Experiment I are shown in Figure 4. From Figure 4, we can observe that Fuzzy algorithm and Fuzzy-Best algorithm can balance the load of servers broadly with fewer tasks, while random algorithm has a great difference load between each server because of uneven task allocation.



a) Random Algorithm



(b) Fuzzy Algorithm

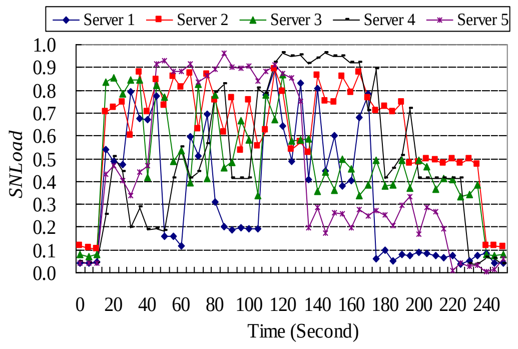


(c) Fuzzy-Best Algorithm

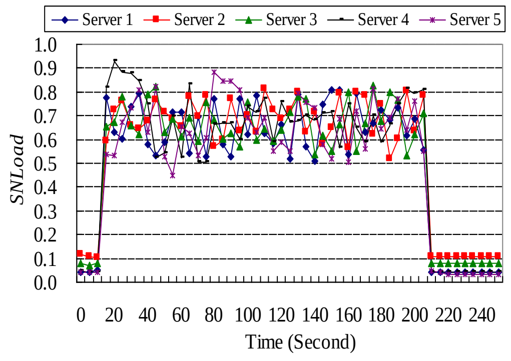
Figure 4. Effects of Task Scheduling Algorithms in Experiment I

The results of Experiment II are shown in Figure 5. From Figure 5, we can find that fuzzy algorithm has a better server load control in task scheduling, and the server load is mostly kept between 0.5 and 0.8. When the server load increase or decrease suddenly, such as the load of Server 1 increases suddenly in the 25s, and the load of Server 5 decreases suddenly in the 55s, fuzzy algorithm can quickly adjust the tasks distribution to make the load of these servers return to steady state.

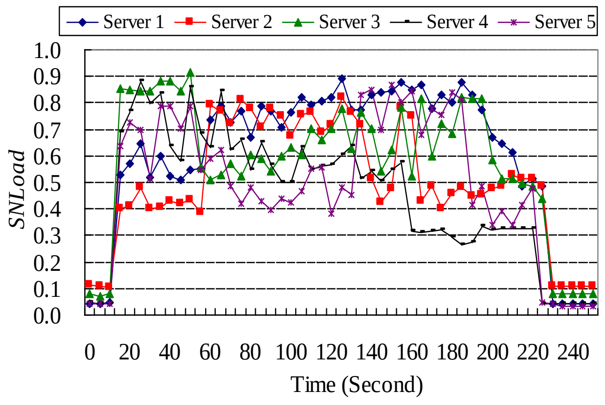
Random algorithm has great load fluctuations of each server and cannot balance the server load effectively. Fuzzy-Best algorithm select the server with the smallest load to execute task scheduling and the running of each task effects the server load differently, which leads to some servers have heavy load in a long time while some servers have a smaller load, such as the load of Server 3 and Server 2 has a large difference in the 15~60s. Meanwhile, some servers are not assigned to execute new task after completing a task, which results in larger bouncing of server load.



(a) Random Algorithm



(b) Fuzzy Algorithm



(c) Fuzzy-Best Algorithm

Figure 5. Effects of Task Scheduling Algorithms in Experiment II

4.2. Execution Time of Tasks

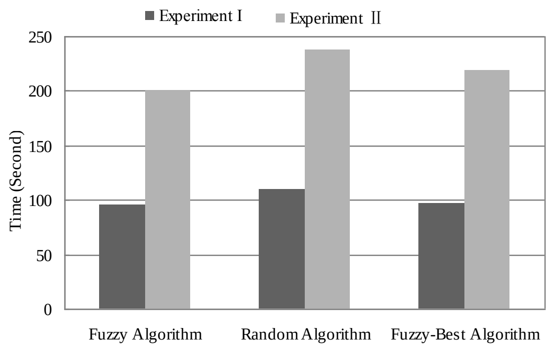


Figure 6. The Time to Complete Tasks of Each Algorithm

From the total execution time of the tasks in Figure 6, we know that Fuzzy algorithm has the minimum time to complete all tasks with 200s, Random algorithm required for the longest time, and the Fuzzy-Best algorithm takes time for 219s in Experiment II. But the execution time of tasks of three algorithms are near in Experiment I.

According to the results of Experiment I and Experiment II, Fuzzy algorithm and Fuzzy-best algorithm can control the server load relatively well with less tasks. When there are many tasks, Fuzzy algorithm can control the server load significantly better with less executing time than Fuzzy-best algorithm.

5. Conclusion

In this paper, we have studied the program information collection in P2P IPTV applications. We proposed a distributed multi-protocol crawler which is used to harvest program information of various P2P IPTV applications. Moreover, we used a task scheduling model based on fuzzy control to improve performance of the crawler. The results show that fuzzy algorithm can balance the nodes load effectively with less task execution time. In the next work, we focus on characteristic analysis and automatic sorting of programs and establishment of IPTV repository.

Acknowledgements

This work was sponsored by the National Key Technology R&D Program (Grant No. 2012BAH18B05) and the Scientific Research Foundation for the Youth Scholars of Sichuan University (Grant No. 2013SCU11017).

References

[1] D. Hughes, J. Walkerdine and K. Lee, "Monitoring challenges and approaches for P2P file-sharing systems", The First International Conference on Internet Surveillance and Protection, Cap Esterel, Cote d Azur, France, **(2006)** August 27-29.

[2] Ipoque Internet study 2008/2009 finds Web and streaming outgrows P2P traffic [EB/OL]. [2011-3-12]. http://www.ipoque.com/en/news-events/press-center/press-releases/2009/ipoque-internet-study-20082009-finds-web-and-streaming.

[3] Wang, Y. Liu, Y. X. Yang and X. Y. Zhou, "Solving the app-level classification problem of P2P traffic via optimized support vector machines", Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, Jinan, China, **(2006)** October 16-18.

[4] P. Eittenberger, U. R. Krieger and H. M. Markoyich, "Measurement and analysis of live-streamed P2PTV traffic", 6th International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks, Zakopane, Poland, **(2010)** January 14-16.

[5] Y. H. Chu, S. G. Rao, S. Seshan and H. Zhang, "A Case for End System Multicast", Proceedings of ACM SIGMETRICS, Santa Clara, CA, USA, **(2000)** June 18-21.

[6] X. Y. Zhang, J. C. Liu, B. Li and T. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL. USA, **(2005)** March 13-17.

[7] W. X. Wang, X. S. Chen and H. Z. Wang, "IPTV-RM: A Resources Monitoring Architecture for P2P IPTV Systems", Journal of Networks, vol. 7, no. 10, **(2012)**, pp. 1624-1630.

[8] H. Z. Wang, X. S. Chen and W. X. Wang, "A measurement study of polluting a large-scale P2P IPTV system", China Communications, vol. 8, no. 2, **(2011)**, pp. 95-102.

[9] H. Z. Wang, X. S. Chen, W. X. Wang and Z. H. Hao, "Understanding pollution dynamics in large-scale peer-to-peer IPTV system", Journal of Central South University, vol. 19, no. 8, **(2012)**, pp. 2203-2217.

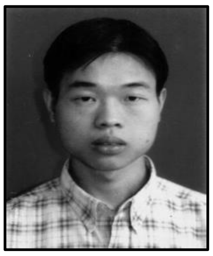
[10] M. Du, X. S. Chen and J. Tan, "A novel P2P traffic identification algorithm based on BPSO and weighted KNN", *China Communications*, vol. 8, no. 2, **(2011)**, pp. 52-58.

[11] S. Agarwal, J. P. Singh and A. Mavlankar, "Performance and quality-of-service analysis of a live P2P video multicast session on the Internet", Proceedings of 16th International Workshop on Quality of Service, Enschede, the Netherlands, **(2008)** June 2-4.

[12] T. Silverston and O. Fourmaux, "Measuring P2P IPTV systems", 17th International Workshop on Network and Operating Systems Support for Digital Audio & Video, Urbana-Champaign, IL, USA, **(2007)** June 4-5.

[13] C. Wu, B. C. Li and S. Q. Zhao, “Characterizing peer-to-peer streaming flows”, IEEE Journal on Selected Areas in Communications, vol. 25, no. 9, **(2007)**, pp. 1-15.
[14] X. J. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, “Insight into PPLive: A measurement study of a large scale P2P IPTV system”, Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW2006, Edinburgh, Scotland, **(2006)** May 23.
[15] X. J. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, “A measurement study of a large-scale P2P IPTV system”, IEEE Transactions on Multimedia, vol. 9, no. 8, **(2007)**, pp. 1672-1687.
[16] C. Wu, B. C. Li and S. Q. Zhao, “Exploring large-scale peer-to-peer live streaming topologies”, ACM Transactions on Multimedia Computer Communication Applications, vol. 4, no. 3, **(2008)**., pp. 1-23.
[17] K. H. Lee, “First course on fuzzy theory and applications”, Springer, Germany, **(2005)**, pp. 137-144.
[18] W. S. Cheng, “Intelligent control theory and application”, Shanghai Jiaotong University Publishing House, Shanghai **(2006)**, pp. 44-45.

Authors



Wenxian Wang, he was born in Jinjiang of Fujian province of China in 1978. He received his Ph.D. degrees from College of Mathematics, Sichuan University, China, in 2012. He is a lecturer of Network and Trusted Computing Institute, Sichuan University. His research interests include peer-to-peer networks, information security and trusted computing.

Xingshu Chen, she was born in 1968. She received her Ph.D. degree from Institute of Information Security at Sichuan University, Chengdu, China, in 2004. She is a professor and Ph.D. supervisor of College of Computer Science. She is currently the director of Network and Trusted Computing Institute (NTCI) and vice director of Information Management Center. Her general research interests include peer-to-peer networks, information security, computer networks and cloud computing.

Haizhou Wang, he was born in 1986. He received his Ph.D. degree from College of Computer Science, Sichuan University, China, in 2014. He awarded Outstanding Graduate Student of Sichuan University twice in 2009 and 2010, and 3rd Prize in 2010 NVIDIA CUDA Collegiate Programming Contest of China. His research interests include peer-to-peer IPTV systems, information security, and network measurement.

Yi Duan, He was born in 1985. He received his B.E. and M.E. degrees from College of Computer Science, Sichuan University, China, in 2010 and 2013. He is currently working at Tencent. His research interests include peer-to-peer networks and information security.