A Priority-based Dynamic Web Requests Scheduling for Web Servers over Content-Centric Networking

Yukai Tu, Xiuquan Qiao, Guoshun Nan, Junliang Chen and Shasha Li
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China,
Email: qiaoxq,tyk,nanguoshun,chjl@bupt.edu.cn
lishasha6464@163.com

***Abstract*—Content-Centric Networking (CCN) facilitates the content distribution for static Web applications due to its in-network caching. However, when it comes to dynamic Web request, the original fair scheduling of Interest packets on the CCN Web server (i.e. without differentiating a new service request and the service requests being processed) increases the mean response time, resulting in a decrease of service performance. To address this problem, we present a Priority-based Dynamic Web Requests Scheduling (called PBDRS) for CCN Web server. We first increase the priority of the subsequent Interest packet requests belonging to dynamic Web requests being processed, based on the first-come-first-served strategy. Then, services with small-size response data are also scheduled to a higher priority, leveraging the shortest remaining processing time (SRPT) scheduling policy. To validate the proposed approach, we implemented the proposed PBDRS mechanism in our existing CCN Web server and conducted the performance evaluation experiments based on a real dataset crawled from three popular dynamic Web sites of China in a trace-driven way. Experimental results indicate that PBDRS outperforms existing fair scheduling approach of Interest packets in terms of the mean response time.**

***Index Terms*—Content-centric networking, web server, dynamic requests scheduling;**

I. Introduction

To address the increasing needs to the Internet, Content-Centric Networking [1] (CCN, currently known as Named Data Networking [2]), was proposed to establish new networking rules for efficient content distribution using content names instead of the host addresses. In CCN, consumers drive the data exchange by emitting Interest packets to routers or the origin server to fetch Content packets, and one Interest packet receives at most one Content packet. The original intention of this design is to achieve flow balance across the network. However, when it comes to dynamic request, unlike the static Web content which can be stored in pervasive in-network caching, the dynamic Web content is often generated dynamically by Web server when it receives the request. Besides, since the dynamic Web contents vary with different user's personalized requests, it's inefficient to be stored by cache. Therefore, the dynamic Web requests must be all sent to the origin Web server. After the CCN Web server generates dynamical response content, the client needs to send multiple Interest packets with different segment numbers to fetch these corresponding Content packets from the CCN Web server. This independent Interest packet-based request processing mechanism is fully different from the socket-based HTTP processing mechanism as shown in Fig. 1.

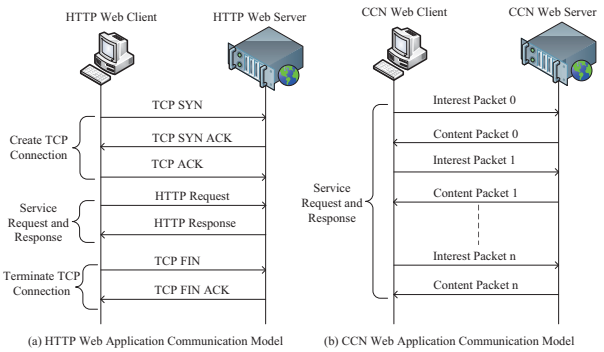


Fig. 1. Different communication models of HTTP Web server and CCN Web server.

In CCN, each Interest packet is an independent request with different names and segment numbers, which can cause the Interest packets of different dynamic Web requests arriving at the server in a different sequence. We divide these Interest packets into two categories: Interest packets belonging to the new requests and Interest packets belonging to the processing requests. Therefore, when both Interest packets arrive at the server and cause congestion, the fair scheduling of Interest packets (i.e. without distinction between two types of Interest packets) will process these requests parallely in the server. When the arrival rate fluctuates greatly over time, the fair scheduling of Interest packets will increase the expected response time. In CCN, given this Interest/Content packets-based dynamic Web request mechanism, we give the priority to subsequent Interest packets to guarantee the FCFS(first-come-first-served) scheduling at the service level in order to decrease expected response delay.

Furthermore, it has been proven that requests requiring short remaining time should be given a higher priority, which is

978-1-4673-9688-2/15 $31.00 © 2015 IEEE
DOI 10.1109/HotWeb.2015.15

beneficial to mean response time, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling policy. In CCN, it is obviously that Web server can obtain the response contents' size of each request after generating corresponding response contents and larger response contents need more Interest packets(i.e. more service time). Server can give these small-size requests a higher priority to reduce the mean response time based on the learned contents' size.

In this paper, we present our Priority-based Dynamic Web Requests Scheduling (PBDRS) by not only giving priority to the Interest packets of requests currently processed, but also giving a higher priority to the subsequent Interest packets belonging to the requests for small-size response contents to reduce the expected mean response time. The remainder of the paper is organized as follows: Section II briefly highlights related work. Section III presents the scheduling problem formulation. The design of our scheduling is displayed in Section IV. Section V shows the experimental results. Finally, Section VI gives the conclusion and future work.

II. Related Work

In this section, we describe the related work about Web application provisioning in CCN and the existing scheduling approaches for HTTP Web servers.

A. Web Application Provisioning Research on CCN/NDN Network

To better support dominant browser-based Web applications in CCN, several applications over CCN have been designed and implemented recently.

For Web client approaches, Shang et al. implemented a JavaScript-based NDN library, trying to exploring an NDN-based Web and a practical need for browser support in NDN research [3] [4]. To natively support the extensive NDN-based Web applications in Web browser, Qiao et al. has developed a complete NDN Web browser (i.e. NDNBrowser) based on open source WebKit. We integrated some inherent NDN features into the browser to support the interactions between Web browser and NDN network [5].

To natively support CCN-based Web applications, for Web server approach, Qiao et al. has designed and implemented a CCN Web server (called CCNxTomcat)[6] based on the CCNx library [7] and Apache Tomcat [8]. However, the previous work on CCN Web server (i.e. CCNxTomcat) only employs a simplified first-come-first-served Interest packets scheduling mechanism and does not differentiate the different Interest packets scheduling.

B. Existing Service Scheduling Approaches for HTTP Web Server

In existing study of queueing theory, it has been proven that give priority to short jobs are optimal with respect to mean response time. In [9], they gave preference to requests for small files or requests with short remaining file size, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling policy [10]. They applied SRPT scheduling to an Apache Web server and the experimental results show that SRPT-based scheduling of connections yields significant reductions in delay at the Web server. McWherter et al. [11] implement and evaluate several different lock scheduling policies for web-driven workloads. Besides the database system scheduling approach, Schroeder et al. propose and implement the methods for scheduling transactions without directly controlling database internal resources in [12], [13]. In this size-based scheduling respect, It has been summarized by Biersack et al. [14] that size-based scheduling can be used in practice to greatly improve mean response times in two important exist applications (i.e. packet scheduling in network routers and connection scheduling in Web servers) without causing unfairness or starvation.

In this paper, our work mainly focuses on the Interest packets-level scheduling issue, by giving preference to the subsequent Interest packets, especially the subsequent Interest packets belonging to request for small-size response content with short processing time of a dynamic Web request.

III. Problem Formulation

Based on the basic scheduling idea mentioned above, in this section, we compare the performance of our proposed PBDRS to the original scheduling mechanism by formulating. we model the whole process of Interest packets by Open Networks of Queues with one service center. The arrival rate of request is λ. The average service time of Interest packets is T_s and there are m threads in the server. Otherwise, the subsequent Interest packets will continue to arrive at the server until the last Interest packet. We denote the time on the round trip of subsequent Interest packets as T_{RTT}.We summarize the notations in TABLE.I.

TABLE I
Notation table

Symbol	Description
λ	Arrival rate of request
T_s	Average service time of Interest packet
m	Threads number in the server
n_m	Minimum request number the server begin scheduling
$\overline{k}$	Average response segment number
R_i	Expected remaining service time of requests i
SR_i	the No.i smallest value in the set of expected remaining service time of request
T_w	Process time of a new request when server with PBDRS
I_1	Expected remaining service time of Interest packets i
SI_i	the No.i smallest value in the set of expected remaining service time of Interest packet
T_{si}	Time between two subsequent Interest packet arrive at server
L_i	Number of Interest packets in the queue in round i
$T_w^{'}$	Process time of a new request when server without PBDRS

Since we apply PBDRS to the server, we give subsequent Interest packets higher priority over first Interest packets. The maximal simultaneous processing requests number is $n_m =$

$\lceil m \cdot \frac{T_{RTT}}{T_s} + m \rceil$, which is also the minimum number of requests the server begin scheduling.

When a client initializes a request, the first Interest packet arrives at the server. Supposing the number of response segment number is k and average response segment number is $\bar{k}$. We discuss the situation when all threads are occupied and there exist n_q Interest packets in the first Interest packet queue and n_t Interest packets on the way. The number of requests waiting to be processed before a new incoming request is $n_f = n_q + n_t + m - n_m$. If $n_f < 0$, the new incoming request doesn't need to wait.

We denote the set $\{R_1, R_2, \ldots, R_{n_m}\}$ as expected remaining service time of requests being processed, and SR_i is the i^{th} smallest value in the set. So the response time of a new request when server with PBDRS T_w:

$$T_w = SR_{n_f \% n_m} + \lfloor \frac{n_f}{n_m} \rfloor \bar{k} T_s + k(T_{RTT} + T_s) \tag{1}$$

It can be observed that, for the Web server with PBDRS, the response time of a new request is closely related to its arrival sequence in the queue. Since we give requests for small response content a higher priority, the remaining service time of requests being processed can be lower to the optimal situation.

If server without PBDRS, we denote the set $\{I_1, I_2, \ldots, I_m\}$ as expected remaining service time of Interest packets being processed and SI_i is the i^{th} smallest value in the set. We discuss the situation when all threads are occupied and there exist $n_q^{'}$ Interest packets in the queue. The time when the first subsequent Interest packet arrive at server:

$$T_{s1} = SI_{n_q^{'} \% m} + \lfloor \frac{n_q^{'}}{m} \rfloor T_s + T_s + T_{RTT} \tag{2}$$

During time T_{s1}, λT_{s1} Interest packets belonging to new requests will arrive at the server. So there are $L_1 = \lambda T_{s1} + n_q^{'} - \varphi_1 - \gamma_1$ Interest packet in front in the queue, where φ_1 is the requests have finished at the last round and γ_1 is the number of Interest packets behind this new incoming Interest being processed during its process time and round trip time. We can calculate that the time between two subsequent Interest packets arrive at server and the number of Interest packets in the queue:

$$T_{si} = SI_{L_{i-1} \% m} + \lfloor \frac{L_{i-1}}{m} \rfloor T_s + T_s + T_{RTT} \tag{3}$$

$$L_{i+1} = \lambda T_{si} + L_i - \varphi_i - \gamma_i \tag{4}$$

Based on Eq.3 and Eq.4, we have the response time of a new request when server without PBDRS $T_w^{'} = \sum_{j=1}^{k} T_{sj}$.

Based on the derivations above, the main advantages of PBDRS is that when the request arrival rate fluctuates greatly over time, the server can greatly reduce the response time of the requests arriving at the server first, result in the reducing of the mean response time. It shows that for the server without PBDRS, the response time will be affected by the request arrival rate and the number of requests being processed. When server employs PBDRS, the number of requests being processed concurrently will be adapted automatically by server and it will adjust the sequence of different request in accordance with response contents' size. Response time of a new dynamic Web request is mainly affected by its arrival sequence.

IV. Implement of PBDRS

In this paper, we apply Priority-based Dynamic Web Requests Scheduling (PBDRS) policy to the existing CCN Web server, i.e. CCNxTomcat [6]. Unlike previous design, we add two new components: CCNQueue and CCNQuery. C-CNQueueu consists of three separated queues: one admits first Interest packets, one admits subsequent Interest packets belonging to request for small-size response content (RFSC) and another contains subsequent Interest packets belonging to request for large-size response content (RFLC). The queue which contains subsequent Interest packets belonging to RFSC has the highest priority. Then is the queue which contains subsequent Interest packets belonging to RFLC. The queue which contains first Interest packets comes last. CCNQuery consists of a hash table, which records the name of requests and the number of response content's segment. Based on the records in the hash table, we calculate average segment number in hash table as the boundary of two subsequent Interest queues. Improved architecture of CCNxTomcat is shown in Fig.2.

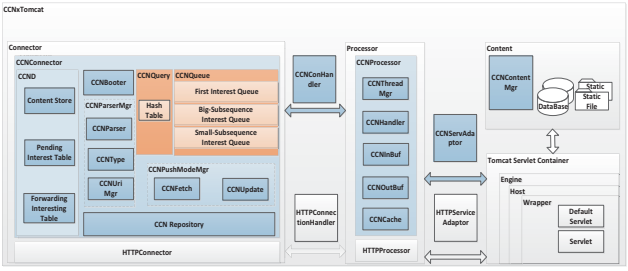


Fig. 2. Improved CCNxTomcat architecture.

The improved CCNxTomcat works as the follow (see Fig.3). Once receiving the Interest packets from CCND(CCN Deamon), the CCNxTomcat takes over the incoming Interest packets as an Interest handler. The static Web requests will be forwarded to the CCN Repository, where stores the static files. The Interest packets of dynamic Web requests will enter the first Interest packets queue in CCNQueue component. If threads are available and both subsequent Interest packets queues are empty, first Interest packets obtain an available thread. Server first determines the types of request(JSP (Java Server Pages) or Servlet). CCNxTomcat will translate these dynamic Web files or services into the corresponding static pages then return the results. A large size generated Web page's will be cut into multiple segments and be stored

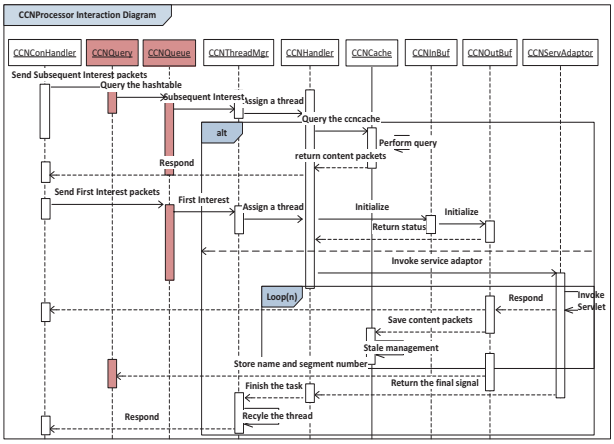
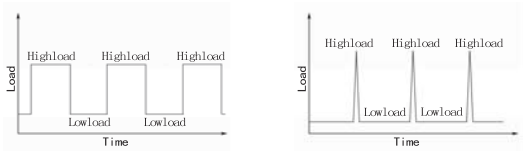


Fig. 3. Handling Process of first and subsequent Interest packets in CCNx-Tomcat.

TABLE II
COMPONENT SPECIFICATION

Component	Specification
CCN Software	CCNx release 0.8.2
CPU of workstation	2 core 2.40Ghz
Memory of workstation	96GB
Memory of virtual machine	8GB
Operating Systems of server and virtual machines	Ubuntu 11.10



(a) Alternating load (b) Intermittent load

Fig. 4. Two different types of transient load

in CCNCache, waiting for the subsequent Interest packets. Finally, server will store the name of the request and the number of corresponding response content's segment into CCNQuery's hash table. If subsequent Interest packets arrive at server, based on the hash table, CCNQuery will calculate average segment number of response contents as the boundary of two subsequent Interest queues, then CCNQuery queries the segment number of corresponding response contents by the name of incoming subsequent Interest packets. If the segment number of corresponding response content less than average segment number, the new incoming subsequent Interest packets will enter the queue for the subsequent Interest packets belonging to RFSC. Otherwise they will enter the queue for the subsequent Interest packets of RFLC. Subsequent Interest packets belonging to RFSC in CCNQueue will be processed first. If the queue for subsequent Interest packets of RFSC is empty, the subsequent Interest packets of RFLC in CCNQueue will be processed. For subsequent Interest packets, server will query CCNCache, return the matched Content packets or discard this Interest packet if matching fails.

V. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

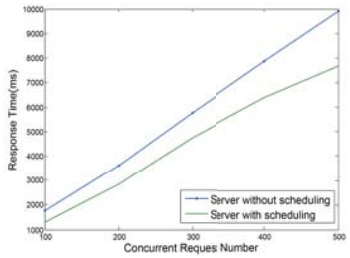
To verify the performance of proposed PBDRS approach, we conducted experiments on a real CCN network testbed comprises two workstations: one carries on five virtual machines; The other workstation works as server. TABLE.II shows more detailed specification of the environment.

In this paper, for each experiment, the mean response time of the dynamic Web requests is concerned about. It's the time from when a client emits the first Interest packets until it receives the last Content packet of this dynamic Web request. We conducted our experiments in different change of load and segment numbers' distribution of response content. We used two types of load: First is alternating load alternates between high load and low load (see Figure.4(a)); The second type called intermittent load, the load is almost always low, but there are occasional "spikes" of high load, evenly spaced apart (see Figure.4(b)).

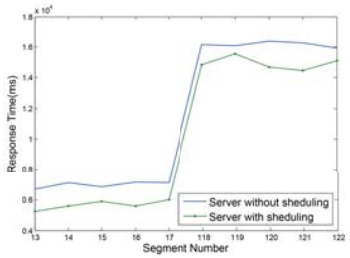
To be closer to the reality, we investigated three popular dynamic Web sites in China, including 12306 (http://www.12306.cn/mormhweb/), Taobao (http://www.taobao.com), and Baidu (http://www.baidu.com). We imitated the users browsing behavior and recorded the return pages' size. Then we transformed the pages' size into the number of CCN Content packets. In general, it's 4KB per packet, and observed the distributions of the packets' number. These three types of dynamic Web service sites can cover most of dynamic Web types in current network. Based on the statistic of these different types of dynamic Web sites, we attempt to observe the practical impact of our PBDRS.

Case 1: As the largest online train ticket booking Website in China, 12306 is response for all online querying and purchases of train tickets in China. The 12306 site releases tickets hourly, which causes the huge traffic in a short time. As a main service of this site, the querying of remaining tickets information is implemented by dynamic Web pages and the response pages' size is almost fixed. We capture some querying result pages by simulating the user behaviors. Then transform the size of each page into the number of Content packets in CCN. Results show that the segment numbers approximate to Gauss distribution with the average value 15 (60KB). In this scenario, we conducted five experiments and each experiment consists six peak periods under the intermittent load. We increased the concurrent request clients from 100 to 500 in peak period by adding 100 clients each time. We use the mean response time as the performance metrics. The results are shown in Fig.5(a)

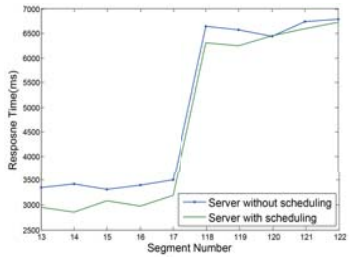
Fig.5(a) shows that when server applies PBDRS, it can effectively lower the mean response time. It offers better service quality to the clients arriving first. For this kind of dynamic Web sites, whose response data's sizes are similar and arrival rate changing greatly, the proposed schedule has a good performance improvement and reducing whole response time by 25.93%.



(a) Mean response time of different concurrent requests for 12306 service scenario under intermittent load



(b) Mean response time of dynamic Web requests with different segment numbers' response pages for Taobao service scenario under intermittent load



(c) Mean response time of dynamic Web requests with different segment numbers' response pages for Taobao service scenario under alternating load

Fig. 5. Results of 12306 and Taobao service scenario

Case 2: Taobao is the largest online C2C shopping Website in China. When shopping online, people often search for a certain commodity by keywords or categories. And all kinds of related commodities will be exhibited in a display page and the detail information of a specific commodity is exhibited in an independent Web page. Since one display page may contain several commodities and people always interested in specific information, the requests for specific commodities pages take the majority of this site's requests. Based on the observation, in our experiments, we mainly focus on two types of dynamic Web pages: one is the display page and the other is the specific information page. Similar to the Case 1, we gathered the response pages from Taobao by simulating user behavior then model their sizes into CCN content segments. Results show that the segment numbers of the display pages of specific commodity approximate to a Gauss distribution with average value 15 (small-size, in this scenario), and the segment numbers of the display pages of all querying commodities approximate to a Gauss distribution with average value 120 (large-size, in this scenario). In this experiment, we generated 500 clients under the intermittent load, and 30% of clients generate the searching request and the rest 70% of clients generate the specific commodity display requests. The experiment results are shown in Fig.5(b). Under the alternating load, we generate requests through the poisson flow. The arrival rate of requests was 0.5 per second at low load and 5 per second at high load. We focus on the response time of different request with different response segment number. The results are shown in Fig.5(c).

Fig.5(b) and Fig.5(c) shows the server with PBDRS is friendly to both requests. Under intermittent load, the PBDRS approach decreases the mean response time by 21.32%. Under the alternating load, server with scheduling decreases the mean response time by 7.94%. Since we give subsequent Interest packets priority over first Interest packets, it has little risk that RFLC will be starved, but we should note that under alternating load, more RFSC, more delay impact on RFLC.

Case 3: Baidu, a search engine Website in China, which is a very typical dynamic Web service in the Internet. Therefore, we also evaluate the performance of CCN Web server with PBDRS for this type of dynamic service. The search engine uses different keywords to search the related content, present dynamical page for clients. We use Baidu Index (http://index.baidu.com) to get the user interest of different keywords. We counted the search index of different keywords for a year (from March, 2014 to February, 2015). Then we related the statistic to the segment number of search response pages' size of different keywords (see Fig.6(a)).

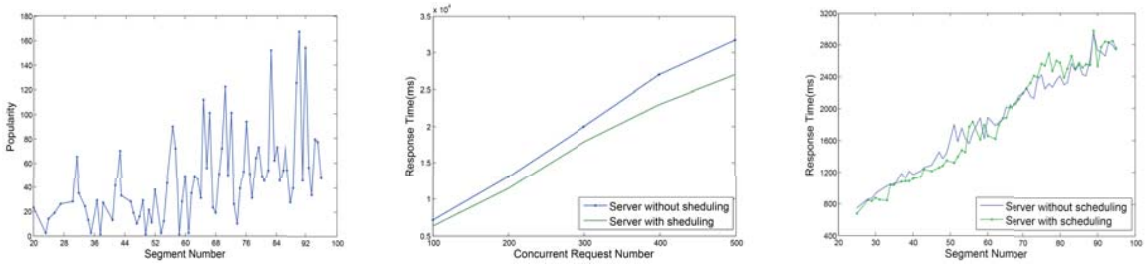
In the experiment, we use the statistic above to generate the client requests. The segment numbers of the searching result pages in accordance with the distribution trend shown in Fig.6(a). We conducted five experiments under intermittent load, and increased the concurrent request clients from 100 to 500 in peak period by adding 100 clients each time. Under alternating load, the arrival rate of requests was 0.5 per second at low load and 5 per second at high load. The results are shown in Fig.6.

Fig.6(b) shows the PBDRS approach can reduce mean response time under intermittent load. Because the results show that the overall mean response time of two scheduling approach is close to each other under alternating load, we give the result that the mean response time of dynamic Web requests with different segment numbers' response pages in Fig.6(c). It shows that there is a little improvement on RFSC. In this experiment, the PBDRS approach saves the mean response time by 14.2% under intermittent load and 1.84% under alternating load.

TABLE III
SAVING RESPONSE TIME BY CCN WEB SERVER WITH PBDRS

Segment Distributions	Intermittent load	Alternating load
12306	25.93%	
Taobao	21.32%	7.94%
Baidu	14.2%	1.84%

We summarize the results of experiments in TABLE.III. Results show that our PBDRS approach can efficiently improve the mean response time of dynamic Web requests for CCN Web server. Since the arrival rate keep changing over the time, PBDRS can reduce mean response time by giving



(a) The relationship of different keywords' popularity with the segment numbers of corresponding searching result pages

(b) Mean response time of different concurrent requests for Baidu service scenario under intermittent load

(c) Mean response time of dynamic Web requests with different segment numbers' response pages for Baidu service scenario under alternating load

Fig. 6. The experiment results for Baidu search application scenario

better service to requests arriving at server first, result in the reducing of mean response time.

VI. Conclusion

In this paper, we consider the problem with the fair scheduling of Interest packets in current Content-Centric Networking (CCN) Web server. Such problem is caused by the one-to-one Interest/Data message exchange principle of CCN protocol which requires multiple Interest packets to fetch multiple corresponding Content packets. All of these requests will be processed simultaneously without effective scheduling, resulting in the performance reduction of the CCN Web server. To address this problem, we propose a Priority-based Dynamic Web Requests Scheduling (PBDRS) for Web servers of CCN. The main idea of PBDRS is to boost the priority of the subsequent Interest packets, especially subsequent Interest packets belonging to RFSC in accordance with the SRPT (Shortest Remaining Processing Time) scheduling theory. We evaluate the performance of PBDRS on a real test bed, using the dataset crawled from three online Web sites in trace-driven fashion. Experimental results show that PBDRS outperforms existing algorithms in terms of the mean response time.

The proposed PBDRS can be further enhanced by dynamically adjusting the weight of different queue so it can reduce the congestion of the RFLC. Pipeline-based mechanism is also left for future work. We will incorporate the pipeline-based mechanism into our scheduling since pipeline can save round trip time, which results in the reduction of mean response time.

Acknowledgment

This work was supported by the National Key Basic Research Program of China (973 Program) under Grant No. 2012CB315802, the National Natural Science Foundation of China under Grants No. 61171102 and No. 61132001, the Prospective Research on Future Networks of Jiangsu Future Networks Innovation Institute under Grant No. BY2013095-4-01, the Beijing Nova Program under Grant No. 2008B50, and the Beijing Higher Education Young Elite Teacher Project under Grant No. YETP0478.The authors would like to express their sincere gratitude for the anonymous reviewers helpful comments.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," *Communications of the ACM*, vol. 55, no. 1, pp. 117–124, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063204

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014. [Online]. Available: http://dx.doi.org/10.1145/2656877.2656887

[3] W. Shang, J. Thompsonz, M. Cherkaoui, J. Burkey, and L. Zhang, "Ndn.js: A javascript client library for named data networking," in *Proceedings of 2013 IEEE Computer Communications Workshops*, ser. INFOCOM WKSHPS'2013. IEEE, 2013, pp. 399–404.

[4] W. Shang, J. Thompson, J. Burke, and L. Zhang, "Development and experimentation with ndn-js, a javascript library for named data networking," Department of Computer Science, Center for Research in Engineering, Media and Performance, UCLA, Tech. Rep. NDN-0014, 2013.

[5] X. Qiao, G. Nan, Y. Peng, L. Guo, J. Chen, Y. Sun, and J. Chen, "Ndnbrowser: An extended web browser for named data networking," *Journal of Network and Computer Applications*, vol. 50, pp. 134–147, 2015.

[6] X. Qiao, G. Nan, W. Tan, L. Guo, J. Chen, W. Quan, and Y. Tu, "Ccnxtomcat: An extended web server for content-centric networking," *Computer Networks*, vol. 75, no. Part A, pp. 276–296, Dec. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614003697

[7] "Ccnx library," 2014, http://www.ccnx.org/releases/ccnx-0.8.0.tar.gz.

[8] "Apache tomcat," 2014, http://tomcat.apache.org/.

[9] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based scheduling to improve web performance," *ACM Transactions on Computer Systems*, vol. 21, no. 2, pp. 207–233, May 2003.

[10] L. E. Schrage and L. W. Miller, "The queue m/g/1 with the shortest remaining processing time discipline," *Operations Research*, vol. 14, no. 4, pp. 670–684, 1996. [Online]. Available: http://www.jstor.org/stable/168730

[11] D. T. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter, "Improving preemptive prioritization via statistical characterization of oltp locking," in *Proceedings of the 21st International Conference on Data Engineering*, ser. ICDE' 2005. IEEE, 2005, pp. 446–457.

[12] B. Schroeder, M. Harchol-Balter, A. Iyengar, and E. Nahum, "Achieving class-based qos for transactional workloads," in *Proceedings of the 22nd International Conference on Data Engineering*, ser. ICDE' 2006. IEEE, 2006, pp. 153–155.

[13] B. Schroeder, M. Harchol-Balter, A. Iyengar, E. Nahum, and A. Wierman, "How to determine a good multi-programming level for external scheduling," in *Proceedings of the 22nd International Conference on Data Engineering*, ser. ICDE' 2006. IEEE, 2006, pp. 60–71.

[14] E. W. Biersack, B. Schroeder, and G. Urvoy-Keller, "Scheduling in practice," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 4, pp. 21–28, 2007. [Online]. Available: http://doi.acm.org/10.1145/1243401.1243407