An Effective Dynamic Service Requests Scheduling Mechanism Specially Designed for CCN Web Servers

XIUQUAN QIAO, YUKAI TU, GUOSHUN NAN, JIWEI HUANG and JUNLIANG CHEN
, Beijing University of Posts and Telecommunications
WEI TAN, IBM T. J. Watson Research Center
SCHAHRAM DUSTDAR, Vienna University of Technology
M. BRIAN BLAKE, University of Miami

The World Wide Web (WWW) has evolved from rendering static pages to rich media with heavy client-server interactions. Content-Centric Networking (CCN), as a named data-based clean-slate future Internet architecture, has totally different protocols and communication patterns compared with the host-to-host IP network. CCN employs a completely different request/response pull communication model of Interest/Content packets to achieve the flow balance in a network. This means that in CCN, when the size of the content generated dynamically by the Web Servers is relatively large, the Web client need send multiple Interest packets with different segment numbers for the same dynamic Web request to fetch the corresponding Content packets. In this novel communication paradigm, we find that the fair scheduling of Interest packets (i.e. without differentiating the new service requests and those requests being processed) will add the average expected response time, especially when the Web requests with large-size response pages dominate or when the requests fluctuate greatly over time. To improve the performance of CCN Web Servers, we present an effective First-Come-First-Served (FCFS) dynamic service requests scheduling scheme specially designed for CCN Web Servers. To validate the proposed approaches, we developed and incorporated these new functions into the existing CCN Web server implementation (named CCNxTomcat), and tested its performance by simulating the workload of three popular dynamic Web sites in China in a CCN testbed. Experimental results show that the improved CCNxTomcat is more friendly to the dynamic Web requests with large-size response pages and outperforms the existing CCNxTomcat in terms of mean response time in most cases.

Categories and Subject Descriptors: D.2.3 [**Network components**]: End nodes

General Terms: Design, Model, Performance, Experimentation

Additional Key Words and Phrases: Content-centric networking, web server, dynamic requests scheduling, web application, named data networking, information-centric networking, future internet

ACM Reference Format:
Xiuquan Qiao, Yukai Tu, Guoshun Nan, Jiwei Huang, Junliang Chen, Wei Tan, Schahram Dustdar, M. Brian Blake. 2015. An Effective Dynamic Service Requests Scheduling Mechanism Specially Designed for

---

This work is supported by the National Key Basic Research Program of China (973 Program) under Grant Nos. 2012CB315802 and 2013CB329102, the National Natural Science Foundation of China under Grants Nos. 61171102 and 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.
Author's addresses: X. Q. Qiao (corresponding author), G. S. Nan, J. W. Huang and J. L. Chen, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications; Y. K. Tu, School of Information and Communication Engineering, Beijing University of Posts and Telecommunication; W. Tan, IBM T. J. Watson Research Center; S. Dustdar, Distributed Systems Group, Vienna University of Technology; M. B. Blake, Graduate School, University of Miami; email:qiaoxq@bupt.edu.cn.

DOI:http://dx.doi.org/10.1145/0000000.0000000

CCN Web Servers. *ACM Trans. Web* 0, 0, Article 0 (2015), 29 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Today's browser/server-based Web has become an important and popular Internet application platform and provides diverse upper-layer applications, including communication, entertainment, social networking, and information retrieval [Gill et al. 2011]. With the rapid growth of content search [Torres et al. 2014], social networking, cloud computing, video sharing [Mitra et al. 2011] and smart phone applications, Web sites have increasingly evolved to dynamic content generation applications to provide dynamic interactions and personalized experiences to the Web sites users. Based on the HTTP protocol and TCP/IP network, the existing HTTP Web servers have been well designed to provide the dynamic Web applications. However, with the evolution of Internet applications, contemporary IP-based Internet architecture increasingly finds itself not capable to meet the demands of current network usage patterns. The host-to-host communication model of existing IP network aims to establish a conversation between exactly two machines. Today, people value the Internet for what content it contains, but communication is still in terms of its origin. Consequently, the existing IP network can hardly address the prevalent technical challenges, such as location-independence, scalable content distribution, mobility and security [Jacobson et al. 2012]. For example, the existing network has to employ the patching approaches to enhance the security of upper-layer applications [Grier et al. 2011].

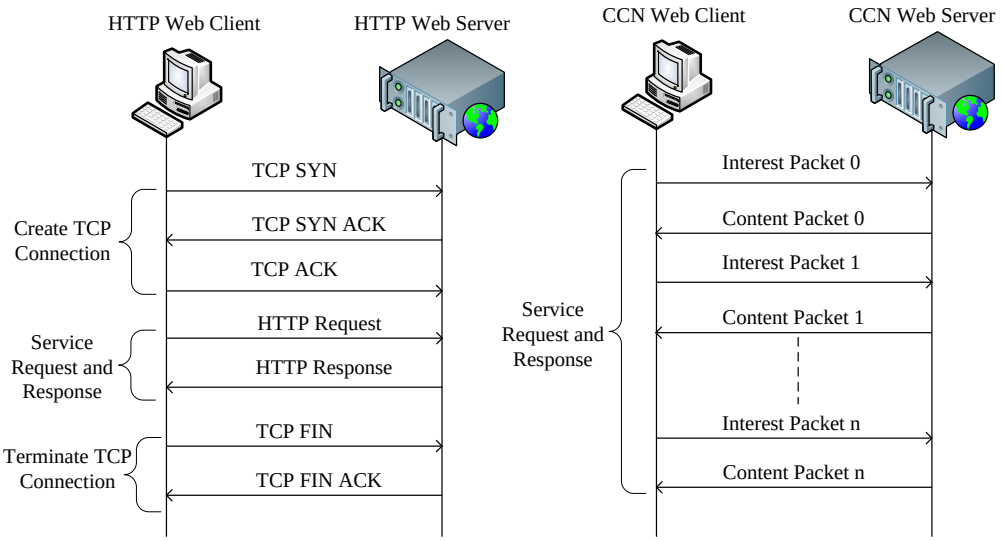
In the quest for solutions that answer the needs of today's and future Internet, Content Centric Networking (CCN), as a new trend in the research community, was proposed to build a novel network architecture that routes content by name and not by the host's network address. This is a really significant improvement, changing the content distribution from where to what. CCN decouples the content from the location and provides the pervasive in-network caching, which enables reusable resources cached in routers closer to users. In CCN, communication is driven by the data consumer completely. Consumers must send Interest packets with the content name directly to routers and fetch Content packets from either intermediate routers' caches or the origin servers. CCN employs a completely different Interest packet/Content packet-based request-response pull communication model to achieve the flow balance in a network. Therefore, naturally, CCN is a typical pull-type network. There are no additional data packets without any solicitation in network. Obviously, CCN has totally different protocols and communication patterns from a host-based TCP/IP network. The CCN's revolutionary architecture has a profound impact on how to provision the upper-layer Web applications. This Interest packet/Content packet-based request/response pull communication model appropriately fits the static Web content request services. Due to the in-network caching, the router nodes can automatically cache the popular content packets. For one visit to a content, maybe a part of Interest packets are satisfied by the intermediate router caches closer to the consumers and other Interest packets are forwarded to a hosting Web server. Therefore, this new Interest packet/Content packet-based communication model can effectively satisfy most of the popular static Web content requests and greatly reduce the redundant network traffic. However, for the dynamic Web request services, this novel Interest/Content packets-based communication model brings some new challenges for CCN Web Servers:

— *Interest/Content Packets-based Dynamic Web Request Mechanism in CCN*. Unlike the static Web content, dynamic Web content is often generated by Web servers when they receive the requests. In addition, since the dynamic Web content often meet the user's personalized requests and cannot be shared by other users. There-

fore, the pervasive in-network caching is not suitable for dynamic and personalized Web content. Thus, the dynamic Web requests must be all routed to the origin Web servers. This means that when the result generated dynamically by the Web Servers consists of multiple content packets, the client needs to send multiple Interest packets with different segment numbers to fetch these corresponding Content packets from Web Servers. Consequently, for the dynamic Web requests, CCN Web servers have to process much more incoming Interest packet requests by comparison with the existing HTTP Web Servers. Compared with the HTTP Web Servers, the CCN Web Servers must accommodate several enhanced features. Specifically, for a dynamic Web request in CCN, the first Interest packet is responsible for triggering the back-end script logic to generate content dynamically, and the subsequent Interest packets only fetch the remaining Content packets from CCN Web Servers when the response data is large than one Content packet. This independent Interest packet-based request processing mechanism is completely different from the socket-based HTTP processing mechanism as shown in Fig. 1. As we can see, in existing HTTP Web Servers, a socket connection is established when the Web client sends the dynamic request. Once receiving the socket creation request, the HTTP Web Servers will allocate a thread to deal with all request and response messages transmitted by this socket until this service is completely executed. However, in a CCN network, each Interest packet is an independent named-based routing request, which maybe results in that the Interest packets of one dynamic Web request arrive at the Interest requesting queue at server side in a different sequences. Therefore, when a large number of Interest packets (including the new service requests and the service requests being processed) arrive at the server side and cause congestion, the fair scheduling of Interest packets (i.e. without differentiating which belongs to the new service requests and which belongs to the service requests being processed) will increase the expected response time, especially when the requests fluctuate greatly over time or when the dynamic Web requests with large-size response pages occupy the request majority of a Web server. Naturally, this Interest/Content packets-based dynamic Web request mechanism raises an issue of whether the subsequent Interest packets of a dynamic Web request being processed deserve a prioritized service to guarantee the first-come-first-served processing at the service level or not in terms of decreasing user-perceived delay.

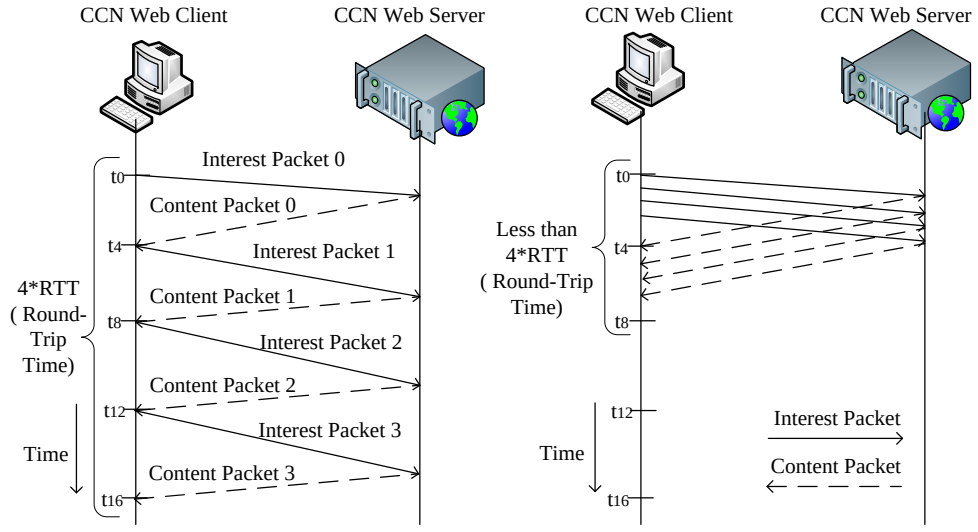
— *Pipeline-based Flow Control Mechanism in CCN*. In CCN, one Interest packet retrieves at most one Content packet. This basic rule ensures the flow balance in the network. To enhance the performance of one-to-one Interest/Content packets exchange, multiple Interest packets may be issued at once. The consumer can also dynamically adjust the pipeline size (i.e. window size) by changing the number of Interests it issues to best fit the link capacity. As shown in Fig. 2, under normal circumstances, the consumer must take several round trips to receive multiple Content packets; however, by pipeline-based flow control, the consumer takes only a little more than one round trip time to receive multiple Content packets. Therefore, for CCN Web Servers, how this pipeline-based flow control mechanism affects the dynamic Web request scheduling and how to employ the pipeline mechanism to further improve the CCN Web server performance also requires exploration and investigation.

It can be observed that CCN's new Interest/Content packet-based request/response mechanism is completely different from existing HTTP/TCP/IP-based communication patterns. Therefore, it's necessary to design the new CCN Web server [Qiao et al. 2014] and CCN Web browser [Qiao et al. 2015] to natively support CCN-based Web applications [Nan et al. 2015]. However, the existing CCN Web server implementation



(a) HTTP Web Application Communication Model (b) CCN Web Application Communication Model

Fig. 1: Different communication models of HTTP Web servers and CCN Web Servers.



(a) CCN Web Application Communication Model Without Pipeline (b) CCN Web Application Communication Model With Pipeline

Fig. 2: CCN communication models with and without Pipeline.

CCNxTomcat [Qiao et al. 2014] does not differentiate the different Interest packets scheduling and only employs a fair First-In-First-Out (FIFO) scheduling of Interest packets. It is well-known from queueing theory that SRPT (Shortest Remaining Processing Time) scheduling minimizes queueing time [Schrage and Miller 1996]. Since the services being processed need shorter remaining processing time than the newly

incoming service requests, the prioritized scheduling of the subsequent Interest packets belonging to the services being processed can facilitate the early arrived dynamic Web requests to be completed as soon as possible, without tolerating more delays due to the completely fair scheduling of Interest packets. Therefore, is it possible to reduce the expected response time of dynamic Web request at a CCN Web server, simply by changing the order in which we schedule the Interest packets? That is the question we answered in this paper. The main contributions of this work can be summarized as follows.

— To the best of our knowledge, this is the first work to explore the Interest/Content packet-based dynamic Web request scheduling approach especially designed for CCN Web Servers, in which, given much faster when Web servers process the services being processed than the newly incoming service requests, we differentiate the first Interest packets and the subsequent Interest packets based on a novel queuing discipline, and prioritize the subsequent Interest packets of services being processed when conditions permit to trade off the user-perceived service delay, called "Interest packet-based Dynamic Service Request Scheduling" (IDSRS).
— To further improve the dynamic Web request performance of CCN Web Servers, for the multiple subsequent Interest packets of one dynamic Web request scenario, we present an adaptive pipeline-based Interest packets processing mechanism by dynamically adjusting the pipeline size (or window size) based on the segment numbers of remaining Content packets. The CCN Web Servers can feedback the fitted pipeline size to the client based on the size of the content it generated, so that the client can reduce the round trips of fetching the remaining content between Web servers, called "Adaptive Pipeline-based Dynamic Service Flow Control" (APDSFC).
— To validate the proposed approaches, we developed and incorporated the dynamic service scheduling components into existing CCN Web server implementation (named CCNxTomcat), and tested the performance by simulating three popular dynamic Web sites in China in a CCN experiment network in a LAN environment. Experimental results show that the improved CCNxTomcat is more friendly to the dynamic Web requests with large-size response pages and outperforms the existing CCN Web server implementation (CCNxTomcat) in most cases.

The remainder of this paper is organized as follows: Section 2 introduces the technical background of CCN network. In Section 3, we review the sate of the art relevant to the Web application provisioning approaches on CCN network and the existing scheduling approaches for HTTP Web servers. Section 4 presents the scheduling problem formulation. Section 5 gives the experimental setup and performance evaluation results. Section 6 discusses the remaining challenges and lessons learned. Section 7 concludes our work and outlines the future research.

2. BACKGROUND: CONTENT-CENTRIC NETWORKING

The existing IP-based Internet architecture designed over fifty years ago, was built upon a host-to-host communication model. Its main design goal is to satisfy the basic Internet usage pattern at that time, i.e. forwarding data packets among a limited number of stationary and trusty machines [Perino and Varvello 2011]. However, with the evolution of Internet and the constantly emerging of new applications, the IP-based Internet architecture has exposed the native defects for some new Internet usage requirements, such as support for scalable content distribution, mobility and security. To address these new challenges, the research community has begun to explore a new clean-slate Internet architecture in recent years. In this context, Content-Centric Networking (CCN) has emerged as a promising candidate for the architecture of the Fu-

ture Internet. CCN (currently known as Named Data Networking, NDN[1], which is further developing the CCN architecture funded by the US Future Internet Architecture program [Zhang et al. 2014].) is a novel networking paradigm which focuses on content dissemination rather than host-to-host connectivity. Its communications architecture is completely built on the named data, not a host address. Thus, CCN does not have the notion of end point information in its traffic packets which contain content names rather than a location address. Essentially, CCN aims to change the thin waist of Internet from IP to named data as shown in Fig. 3 [Jacobson et al. 2012].

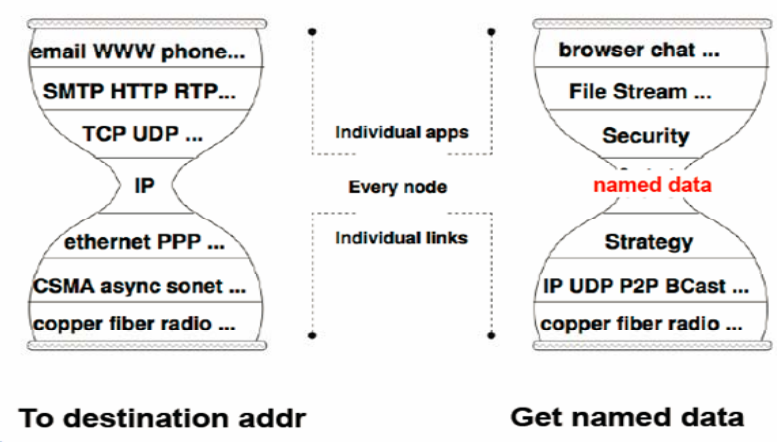


Fig. 3: The Internet architecture shift from IP to CCN.

This section provides a high level introduction of the specific principles of CCN network. CCN offers two distinct packet types in the network: *Interest packet* and *Content packet*. Both packet types carry a name, which uniquely identifies a content segment. Each packet is routed and forwarded based on its name independently. In a CCN network, the large content that cannot be carried in a single *Content packet* are segmented into multiple packets with different segment numbers. In general, clients send *Interest packets* with the requested content names. The *Interest packets* are routed and forwarded based on the name-based routes toward data sources. The in-network caching or origin content server will reply with *Content packets* containing the same name [Yuan and Crowley 2013].

Specifically, each CCN router maintains three major components as shown in Fig. 4 [Jacobson et al. 2009]: the Forwarding Information Base (FIB), the Pending Interest Table (PIT), and the Content Store (CS). The FIB is responsible for deciding where *Interest packets* can be forwarded by longest prefix name matching. The PIT maintains the records for unsatisfied *Interest packets*. Note that each PIT entry contains one or multiple incoming faces: multiple incoming faces indicate that the same *Content packet* is requested from multiple downstream users. The CS is a temporary buffer for *Content packets* that pass through. When the consumer sends out an *Interest packet*, which carries a name that identifies the desired data, the CCN router looks up its CS to find out whether it has a cached copy of the desired content. If so, the router responds with the cached copy. Otherwise, it records the interface from which the *Interest packets* comes into the PIT, and forwards the *Interest packets* by FIB. The router

[1]NDN, http://named-data.net/

stores the *Interest packets* in its PIT along with the interface from which the packet has been received, until the expected data is received. When the router receives more than one *Interest packet* for the same data, it only forwards the first packet upstream towards the data source, and other incoming *Interest packets* are recorded in the existing PIT entry. When the *Interest packets* eventually arrive at a node (i.e. intermediate caching node or origin content server) with the desired data, the *Content packets* will be returned by the reverse path left by the *Interest packets* [Yuan et al. 2012]. It is worth noting that one *Interest packet* can only retrieve at most one *Content packet*. Therefore, there are no unsolicited *Content packets* in CCN. This symmetry between *Interest* and *Content packets* guarantees the flow balance in a CCN network. In addition, the pervasive in-network caching function of CCN facilitates the consumer to find a content copy closest to the client, which greatly reduces the network traffic and the response delay.

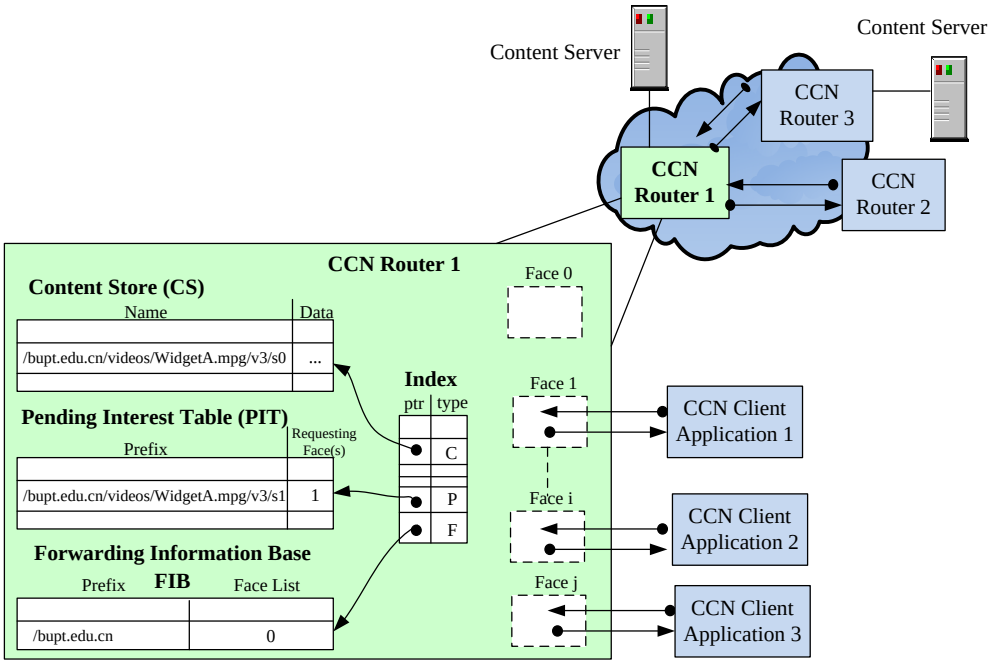


Fig. 4: The inherent principles of CCN network.

3. RELATED WORK

In this section, we review the state-of-the-art techniques with relevance to the Web application provisioning approaches on CCN/NDN network, pipeline-based flow control mechanism and the existing scheduling approaches for HTTP Web servers.

3.1. Web Application Provisioning Research on CCN/NDN Network

As mentioned in [Wang et al. 2012b], S. Wang et al. argued that "the success of CCN/NDN would largely depend on whether the new architecture can support various application needs more effectively and efficiently as it promises. So, designing applications over CCN/NDN is an extremely important issue to solve". Today, Web-based

applications have become more and more popular. The browser/server-based Web application architecture has evolved into an effective Internet service platform for various applications, such as e-business, online gaming, online ticketing, search service, social networking service and cloud service. Therefore, how to effectively provide the dominant browser-based Web applications is an important and meaningful issue for any clean-slate Future Internet architecture including CCN/NDN [Qiao et al. 2014]. Much research has been done to bridge the existing Web applications and CCN/NDN network. In the following, we review the state-of-the-art techniques with relevance to the provisioning of Web applications on CCN/NDN networks.

(1) *HTTP-CCN Protocol Conversion Gateway/Proxy Approaches.* Given that: a) The clean-slate CCN needs the relatively large-scale test and real traffic to prove its feasibility and performance. b) As a clean-slate future Internet architecture, how to coexist with existing network and realize the smooth transition of CCN is still an open issue. c) Extensive effort is still required to determine how CCN's new features meet the application needs and how to interact with pre-existing content identification mechanism (e.g. HTTP content description), Wang et al. [Wang et al. 2012a] designed and implemented a HTTP-CCN gateway to transform HTTP request and HTTP response messages into CCN Interest and Data packets. Similarly, the PARC laboratory team also developed an application called HttpProxy [CCNx library 2014] and integrated it with CCN prototype (i.e. CCNx library), which implements a protocol conversion between CCNx and HTTP protocol. The gateway/proxy approach maintains the existing HTTP Web clients and HTTP Web servers unchanged. Therefore, it can be easily used to build CCN testbed with real HTTP traffic, which will help to validate the specific mechanisms of CCN and to find a smooth transition way from existing IP network to CCN. However, due to the protocol conversion cost for four times between CCNx and HTTP for each pair of request and response, as well as the protocol mismatching issue between HTTP and CCNx protocols, the HTTP-CCN gateway/proxy approach is not very efficient.
(2) *Web Client Approaches.* The Web browser is an important client tool to fetch and present Web pages visually, which are mainly built on Microsoft Windows, iOS and Android platforms currently. However, the CCNx package (i.e. CCN prototype) cannot support the Microsoft Windows and iOS platforms. Furthermore, the installation and configuration of CCNx package has a high technical threshold for the general application developers, which requires the deep understanding of the low-level communication protocol details. Therefore, there is an obvious gap between CCN network and Web front-end application. To explore the high-level goal of an NDN-based Web and a practical need for browser support in NDN research, Shang et al. [Shang et al. 2013b] [Shang et al. 2013a] developed a pure JavaScript NDN library (i.e. NDN.JS). Compared with the HTTP-CCN proxy approach, the NDN.JS approach tries to import the NDN function into the Web browser with the JavaScript library. This approach makes the Web browser identify NDN naming scheme in part of HTML Tags. NDN.JS has the advantage of working with mainstream Web browsers easily. In that case, the impact of NDN.JS on existing browsers is relatively less. However, since the kernel of existing Web browsers cannot support the NDN mechanism natively, some functions are limited. For example, how to render NDN ContentObjcets in the Web browser is still an issue to be addressed by NDN.JS approach. To enable the Web browser natively support the extensive NDN-based Web applications, our research team has developed a complete NDN Web browser (called NDNBrowser) based on open source WebKit. Some inherent NDN features are seamlessly integrated into the browser to support the direct interactions between Web browser and NDN network [Qiao et al. 2015].

(3) *Web Server Approach*. In browser/server-based Web application architecture, Web servers play a key role in satisfying the needs of a large and growing community of Web users. However, the existing Web servers are designed for the hypertext transfer protocol (HTTP) and socket-based host-to-host communication model rather than CCN, and therefore cannot communicate with CCN routers directly. To address this issue, according to the novel features of CCN, our research team has designed and implemented a CCN Web server (called CCNxTomcat) based on the CCNx library [CCNx library 2014] and Apache Tomcat [Apache Tomcat 2014], to natively support CCN-based Web applications [Qiao et al. 2014]. The native CCN functions are integrated into Apache Tomcat to support the seamless interactions between Web servers and CCN routers. CCNxTomcat implemented the Interest packets-based processing mechanisms for both static and dynamic content resource requests. However, our previous CCN Web server (i.e. CCNxTomcat) only employs a basic First-Come-First-Served Interest packets scheduling mechanism and does not differentiate the different Interest packets scheduling. Furthermore, how the pipeline-based Interest packets flow control mechanism affects the performance of CCN Web Servers is also not explored. In this article, we firstly propose to explore the differentiated scheduling issue of Interest packet requests and the adaptive pipeline-based Interest packet flow control issue.
(4) *Communication Patterns for Web Interaction in CCN/NDN Network*. CCN/NDN employs a pure pull-based communication model and the Interest packets hardly consider the carrying issue of user data. However, in modern Web communication patterns, the Web users are often both content consumers and content producers. Therefore, the Web clients often need deliver the user-generated content or application state to Web servers. Therefore, how to effectively support the typical push-based Web communication patterns in a CCN/NDN network needs to make further explorations. To this end, Moiseenko et al. [Moiseenko et al. 2014] explored multiple different approaches, and discussed the advantages and drawbacks of each proposed approaches. The primary goal of this research work focuses on a discussion about how NDN can support modern Web communication patterns effectively. In addition, Detti et al. [Detti et al. 2012] considered the design issue of an Information Centric Network (ICN) that uses a routing-by-name architecture, i.e. content requests are routed on the base of the content name by using a name-based routing table. They focused on the scenario of fetching Web contents, assuming to use ICN in place of traditional TCP/IP means. In this scenario they need to handle tens of billions of name-based routes, due to the high numbers of Web contents and to the limited aggregability of their names.

3.2. Pipeline-based Flow Control in CCN/NDN

In a CCN/NDN network, each Interest packet is routed independently based on its content name and one Interest packet retrieves at most one Content packet. This basic rule ensures the flow balance in the network. However, for a large content containing dozens or even hundreds of data packets, the sequential one-to-one Interest/Content packets exchange by sending one Interest packet at once is not an efficient way. Therefore, Jacobson et al. [Jacobson et al. 2009] proposed the pipeline-based flow control to enhance the one-to-one Interest/Data exchange by sending multiple Interests at once. Rozhnova et al. [Rozhnova and Fdida 2012] and Carofiglio et al. [Carofiglio et al. 2012] extended the pipeline-based flow control by dynamically adjusting the pipeline size to best fit the link capacity. To maximize the overall link utilization when the uplink is congested (or narrowband) while the Data downlink is not, Byun et al. [Byun et al. 2013] proposed an adaptive flow control mechanism where multiple Data messages per Interest are allowed in an adaptive manner without violating the operational princi-

ple of original CCN protocol. It can be observed that all existing work focuses on the flow control or flow shaping of CCN router or client sides. But, how the pipeline-based flow control mechanism affects the performance of CCN Web Servers has not yet been explored so far.

3.3. Existing Service Scheduling Approaches for HTTP Web Servers

The differentiating and scheduling of Web requests plays an important role in improving QoS of existing HTTP Web Servers. With the thriving of the Web, much service scheduling research has been done in terms of HTTP Web server performance optimization, many different approaches have been proposed.

At the early stage of the Web, most practical Web servers use a queuing policy based on the Best Effort model, which employs the First-in-First-out (FIFO) scheduling rule to prioritize Web requests in a single queue, and cannot provide Quality of Service (QoS) as described in [Ye et al. 2005]. Given that different Web requests may have different priorities, [Bhatti and Friedrich 1999] discusses classifying Web requests into high, medium, and low priorities using such information as IP addresses and requested Web sites. The Best Effort model does not support the differentiation of requests based on their priority. Instead, requests are treated fairly based on their arrival time. In [Ye et al. 2005], Ye et al. present Web server QoS models that use only one queue for all Web requests, along with several scheduling rules from production planning in the manufacturing domain, to differentiate QoS for classes of Web service requests with different priorities. While the differentiation approach usually offers better service to premium clients, it does not provide any guarantees on the service. Hence, Lu et al. [Lu et al. 2001] presents the design, implementation, and evaluation of an adaptive architecture to provide relative delay guarantees for different service classes on Web Serverss under HTTP 1.1. They use a feedback control method to address the relative delay or jitter in a Web server for assuring QoS. In addition, Wei and Blake [Wei and Blake 2014] use an agent-based, collaborative approach to schedule virtual machines in clouds - a related scenario for scheduling.

In queueing theory, it has been proven in the 1960's that size-based scheduling policies that give priority to short jobs are optimal with respect to mean response time. Therefore, is it possible to reduce the expected response time of every request at a Web server, simply by changing the order of the Web requests? The authors of [Harchol-Balter et al. 2003] well answer this question in practice. They propose a method for improving the performance of Web servers servicing static HTTP requests, which gives preference to requests for small files or requests with short remaining file size, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling policy. They implement SRPT scheduling in an Apache Web server and the experimental results indicate that SRPT-based scheduling of connections yields significant reductions in delay at the Web Servers. Similarly, one try to apply the same techniques to improve user-perceived performance for dynamic Web requests whose responses are created dynamically. However, the size-based scheduling is more difficulty for the dynamic Web requests than the static Web requests. Because the processing of dynamic Web requests involves the database backend, i.e. applying the size-based scheduling idea to dynamic requests means scheduling database transactions to give priority to short transactions. However, it's very difficult to know the processing time of a transaction before actually running it. In addition, the general database systems do not support effective transaction prioritization for Web-based transactional workloads. In order to address these challenges, McWherter et al. [McWherter et al. 2005] implement and evaluate a number of different lock scheduling policies for Web-driven workloads . Besides the scheduling of database system approach, Schroeder et al. propose and implement the methods for scheduling transactions, without directly controlling

database internal resources in [Schroeder et al. 2006a; Schroeder et al. 2006b]. Surprisingly, their experimental results show that external transaction scheduling, when done right, can be as effective as internal (lock) scheduling, without introducing any negative side-effects, such as reduced system throughput or increased overall response times. In this size-based scheduling respect, Biersack et al. [Biersack et al. 2007] summarize that size-based scheduling can be used in practice to greatly improve mean response times in two important real world applications (i.e. connection scheduling in Web servers and packet scheduling in network routers) without causing unfairness or starvation.

With the explosive use of Internet, overloads are of more serious concerns for QoS-aware Web servers. Chen and Mohapatra [Chen and Mohapatra 2003] exploit the dependence among session-based requests and propose a dynamic weighted fair sharing (DWFS) scheduling algorithm to prevent and control overloads in Web servers by utilizing session-based relationship among HTTP requests. DWFS is distinguished from other scheduling algorithms in the sense that it aims to avoid processing of requests that belong to sessions that are likely to be aborted in the near future. Schroeder and Harchol-Balter [Schroeder and Harchol-Balter 2006] provides a detailed implementation study on the behavior of Web servers that serve static requests where the load fluctuates over time (transient overload). They find that performance can be dramatically improved via a kernel-level modification to the Web Servers to change the scheduling policy at the server from the standard FAIR (processor-sharing) scheduling to SRPT (shortest-remaining-processing-time) scheduling. They find that SRPT scheduling induces no penalties. In particular, throughput is not sacrificed and requests for long files experience only negligibly higher response times under SRPT than they did under the original FAIR scheduling.

In addition, given the hardware performance and the processing strategy of dynamic requests greatly affect the performance of Websites, the authors of [You and Zhao 2012] exploit the performance of multi-core CPUs by means of the hard affinity method in the O/S and present a new dynamic request scheduling approach to schedule the dynamic requests based on a weighted-fair-queuing (WFQ) system, according to the distribution of the dynamic request service time.

Obviously, our work is different from the previous work on HTTP Web servers, since the native communication patterns of CCN are completely different from the exiting TCP/IP network and the above HTTP protocol. Specifically, the major difference between our work and existing work is that, in this paper, our work generally focuses on the Interest packets-level scheduling issue (i.e. giving preference to the subsequent Interest packets of services being processed with short processing time of a dynamic Web request) to prioritize the services being processed.

4. PROBLEM FORMULATION

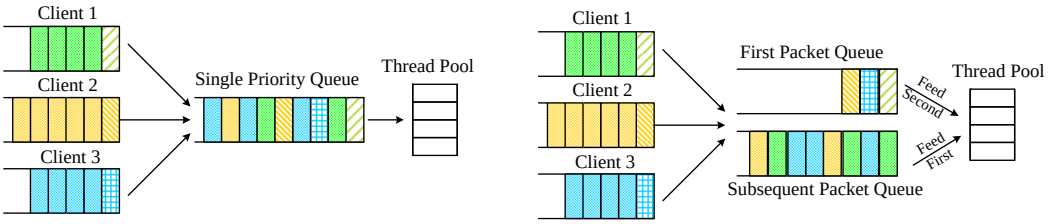
In this section, we introduce the issue of Interest packet-based Dynamic Service Request Scheduling (IDSRS) in CCN Web Servers, and formulate this scheduling model. Please refer to Table I for a summary of the notations we will use throughout the paper.

4.1. Basic Scheduling Idea

To improve the CCN-based dynamic Web performance, in this paper, we attempt to prioritize the service requests being processed (not the newly incoming service requests) by giving higher priority to the subsequent Interest packets of the dynamic Web requests being processed. Figure 5(a) shows the general processing mechanism in existing CCNxTomcat implementation, which employs a fair Fist-In-First-Out (FIFO) scheduling model for all incoming Interest packets. It means that, all the Interest packets including the first Interest packets and the subsequent Interest packets sent

Table I: Notation table

Symbol	Description
λ	Arrival rate of request
T_s	Average service time of Interest packet
m	Threads' number in thread pool
T_d	The time server allocating an available thread
n_m	Minimum request number the server begin scheduling
n_t	request on the way to and from the client
$\overline{k}$	Average response segment number
R_i	Expected remaining service time of requests i
SR_i	the i^{th} smallest value in the set of expected remaining service time of request
T_w	Process time of a new request when server with IDSRS
I_1	Expected remaining service time of Interest packets i
SI_i	the i^{th} smallest value in the set of expected remaining service time of Interest packet
φ_i	Number of finished request at round i
T_{si}	Time between two subsequent Interest packets 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 IDSRS



(a) First-Come-First-Served Fair Model (b) Subsequent Interest Packet Prioritized Model

Fig. 5: Two different scheduling approaches.

by the content consumers enter into a single priority queue, and the server fetches the Interest packet from this queue in a FIFO way, and dispatches a thread to process it. The first Interest packets (representing the newly incoming service requests) and the subsequent Interest packets (representing the existing service requests being processed) are handled fairly. Given that the service requests being processed take less time than the newly incoming service requests, according to the SRPT (Shortest Remaining Processing Time) scheduling principle, we build two priority queues as shown in Figure 5(b): one is the first Interest packets queue, the other is the subsequent Interest packets queue. The latter has higher priority over the former. This scheduling policy guarantees that the subsequent Interest packets of the existing service requests being processed can be handled preferentially. The goal of this scheduling policy is to

reduce the mean user-perceived service delay. In this paper, we are concerned with the response time, which is defined to be the time from when the client sends out the first Interest packets to trigger a back-end dynamic Web request execution until it receives the last Content packet of this dynamic Web request.

According to the different processing mechanism of Interest packets, here, we put the Interest packets into two categories: the first Interest packets and the subsequent Interest packets. When the client initiates a dynamic Web request, it sends out the first Interest packet to server to generate the Web contents dynamically. If the response data's size is larger than a Content packet's size, the result will be divided into multiple Content packets which need multiple Interest packets to fetch. Since one Interest packet returns at most one Content packet in CCN, in most cases, a complete dynamic Web request often contains multiple Interest packets to fetch the corresponding Content packets based on the size of the response data. After all Content packets are fetched from server, this dynamic Web request is completed.

Our main goal is to model the mean response time improvement of the CCN Web Servers with IDSRS (Interest packet-based Dynamic Service Request Scheduling). Based on the basic scheduling idea mentioned above, we model the whole process of Interest packets by Open Networks of Queues with one service center as shown in Fig.6. We denote 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. Without the subsequent Interest packets, a dynamic Web request will end. 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}.

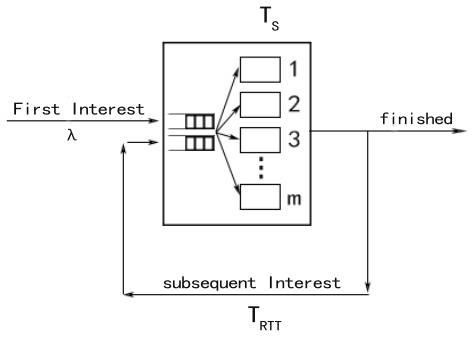


Fig. 6: The networks of queues model of CCN Web Servers processing the Interest packets.

To understand the impact of IDSRS on the performance of CCNWeb Servers, we model the response delays for two cases (server with IDSRS or not) and analyze them.

4.2. Case 1: Server with IDSRS

For the server with IDSRS, we give the subsequent Interest packets priority over the first Interest packets. Besides the Interest packets being processed in the server, the subsequent Interest packets on the trip mean that the dynamic Web requests which they belong to are also being processed by server. We denote the $\overline{T_{RTT}}$ as the mean round trip time. The maximum number of simultaneous processing requests is:

$$n_m = \lceil m \cdot \frac{\overline{T_{RTT}}}{T_s} + m \rceil \tag{1}$$

Where n_m is also the minimum request number over which the server need begin scheduling.

When a client initiates a request, the first Interest packet arrives at the server. Supposing the segment number of response data is k and the average segment number of response data is $\overline{k}$, 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, where $(n_q + n_t + m) > n_m$. So the number of requests waiting to be processed before a new incoming request is:

$$n_f = n_q + n_t + m - n_m \tag{2}$$

We denote the set $\{R_1, R_2, \ldots, R_{n_m}\}$ as the expected remaining service time of requests being processed, and SR_i is the i^{th} smallest value in the set. If $n_q + n_t + m < n_m$, the new incoming request doesn't need to wait. If $n_f > n_m$, besides the remaining service time, the extra waiting time should be added. The response time T_w of a new request for the server with IDSRS is:

$$T_w = \lfloor \frac{n_f}{n_m} \rfloor \overline{k} T_s + SR_{n_f \% n_m} + k(\overline{T_{RTT}} + T_s) \tag{3}$$

Where $\lfloor \frac{n_f}{n_m} \rfloor \overline{k} T_s + SR_{n_f \% n_m}$ means the waiting time before the request begins to be processed, $k(\overline{T_{RTT}} + T_s)$ is the processing time of the request itself. When $0 < n_f < n_m$, the Eq.3 can be further simplified as:

$$T_w = SR_{n_f} + k(\overline{T_{RTT}} + T_s) \tag{4}$$

It can be observed that, for the Web server with IDSRS, the response time of a new request is closely related to its arrival sequence in the queue.

4.3. Case 2: Server without IDSRS

For the server without IDSRS, we denote the set $\{I_1, I_2, \ldots, I_m\}$ as the 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 arrives at the server is:

$$T_{s1} = \lfloor \frac{n_q^{'}}{m} \rfloor T_s + SI_{n_q^{'} \% m} + T_s + \overline{T_{RTT}}. \tag{5}$$

Where $\lfloor \frac{n_q^{'}}{m} \rfloor T_s + SI_{n_q^{'} \% m}$ represents the waiting time of the first Interest packet. When the first subsequent Interest packet arrives at the server, there are λT_{s1} Interest packets belong to the new requests in the queue. The number of Interest packets in the queue is:

$$L_1 = \lambda T_{s1} + n_q^{'} - \varphi_1 - \gamma_1 \tag{6}$$

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 the time between two subsequent Interest packets arriving at the server:

$$T_{si} = \lfloor \frac{L_{i-1}}{m} \rfloor T_s + SI_{L_{i-1}\%m} + T_s + \overline{T_{RTT}}. \tag{7}$$

And the number of Interest packets in the queue:

$$L_{i+1} = \lambda T_{si} + L_i - \varphi_i - \gamma_i \tag{8}$$

Based on Eq.7 and Eq.8, for the server without IDSRS, we have the response time of a new request with k response segments $T_w^{'}$:

$$T_w^{'} = \sum_{j=1}^{k} T_{sj} \tag{9}$$

4.4. Analysis of Two Cases

Based on the deduction above, it shows that for the server without IDSRS, the response time will be affected by the request arrival rate. When the server employs IDSRS, the response time of a new dynamic Web request will be mainly affected by its arrival sequence in the queue. This means that when the request arrival rate fluctuates greatly over time, the server with IDSRS can greatly reduce the response time of the requests firstly arriving at the server, which can reduce the mean response time.

Also, it can be observed from Eq.3 and Eq.9 that when the response segment number k of dynamic Web request is small, for the server without IDSRS, the request will only suffer some remaining service time of Interest packets arriving earlier. However, for the server with IDSRS, the new incoming request has to suffer some remaining service time of Web requests (not the Interest packet level) being processed. That is to say, the IDSRS approach benefits the dynamic Web requests with large-size response pages and the dynamic Web requests with small-size response pages will suffer much delay than the existing fair-scheduling approach for Interest packets. Therefore, whether using IDSRS or not depends on the specific service features of dynamic Web sites.

5. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

To verify our proposed approach, we improved the existing CCNxTomcat to support the IDSRS scheduling policy. In the following subsections, we will show the architecture of improved CCNxTomcat and the detailed processing flowchart. After that, The performance comparisons of the server with IDSRS or not in different request rates and different response data size distribution will be given. At last, we will demonstrate the experiment effect if we apply the pipeline to dynamic Web request in CCN Web server.

5.1. Improved CCNxTomcat

In this paper, we apply our proposed IDSRS scheduling policy to the existing implementation of CCN Web server, i.e. CCNxTomcat [Qiao et al. 2014]. Differently from the origin design, we added a new component: CCNQueue, which consists of two separated queues: one admits first Interest packets and the other deals with subsequent Interest packets. The subsequent Interest packets queue has full priority over the first Interest packets queue. The improved architecture of CCNxTomcat is shown in Fig.7.

After employing IDSRS scheduling policy, the improved CCNxTomcat works as follows (see Fig.8). Upon receiving the Interest packets from CCN network, the CCND component (CCN Deamon) will query each handle to find the proper receiver. The C-CNxTomcat will parse the incoming Interest packets. The static Web requests will be

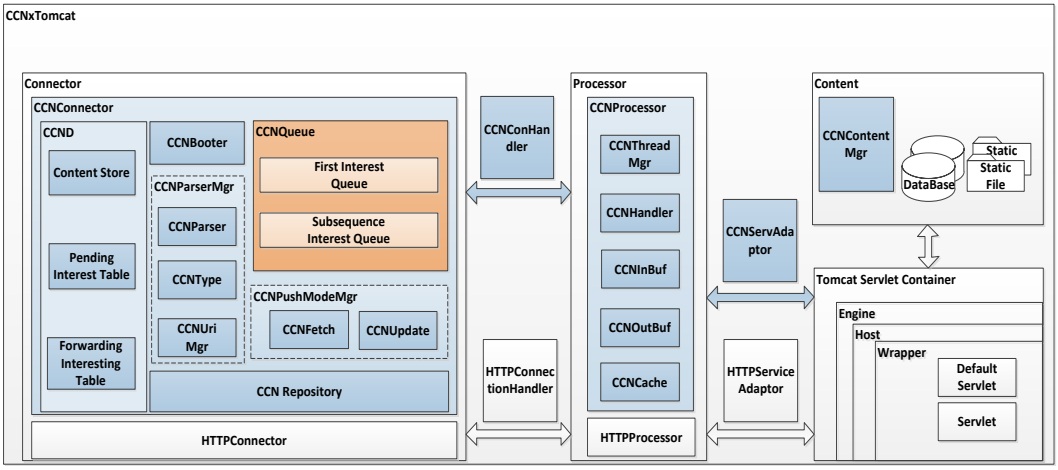


Fig. 7: Improved CCNxTomcat architecture.

forwarded to the CCN Repository component, where stores the static files. The Interest packets of dynamic Web requests will enter the CCNQueue component, waiting for further disposal. If threads are available, the subsequent Interest packets in CCNQueue will be processed first. Server will look up CCNCache for matched Content packets, return the matched Content packets or discard this Interest packet if match fails. If the subsequent Interest packets queue is empty, first Interest packets obtain the available threads. Server will first determine the types of dynamic Web requests, JSP (Java Server Pages) or Servlet, two types of dynamic files in CCNxTomcat, which have similar processing mechanism but different Interest packet names. In the following experiments, the Web clients emit first Interest packets named ccnx:/seg$n$test$m$.jsp, which means the target file is a jsp format file and the segment number of response data is n. Then CCNxTomcat will translate these target dynamic Web requests or services into the corresponding static pages and return them to clients. If the dynamically generated Web page's size is larger than one Content packet's size, it will be cut into multiple content segments and stored in CCNCache, waiting for the subsequent Interest packets requests.

5.2. Performance Evaluation of CCN Web Server With IDSRS

To verify the effectiveness of proposed IDSRS approach, we conduct experiments on a real CCN network testbed to compare the performance of CCNxTomcat with IDSRS scheduling policy or not. In the experiments, we use two workstations: one carried on five virtual machines and each machine generates most 150 clients; The other works as server. TABLE.II shows the detailed hardware and software specifications of the experiment in a LAN environment.

In this paper, for each experiment, we mainly evaluate the mean response time of the dynamic Web requests. The response time of a dynamic Web request is the time from when the client sends out the first Interest packets to trigger a service execution until it receives the last Content packet of this dynamic Web request. In our later experiments, we conducted the experiments in different request modes and segment numbers' distribution of response Web pages. As we stated above, the IDSRS approach

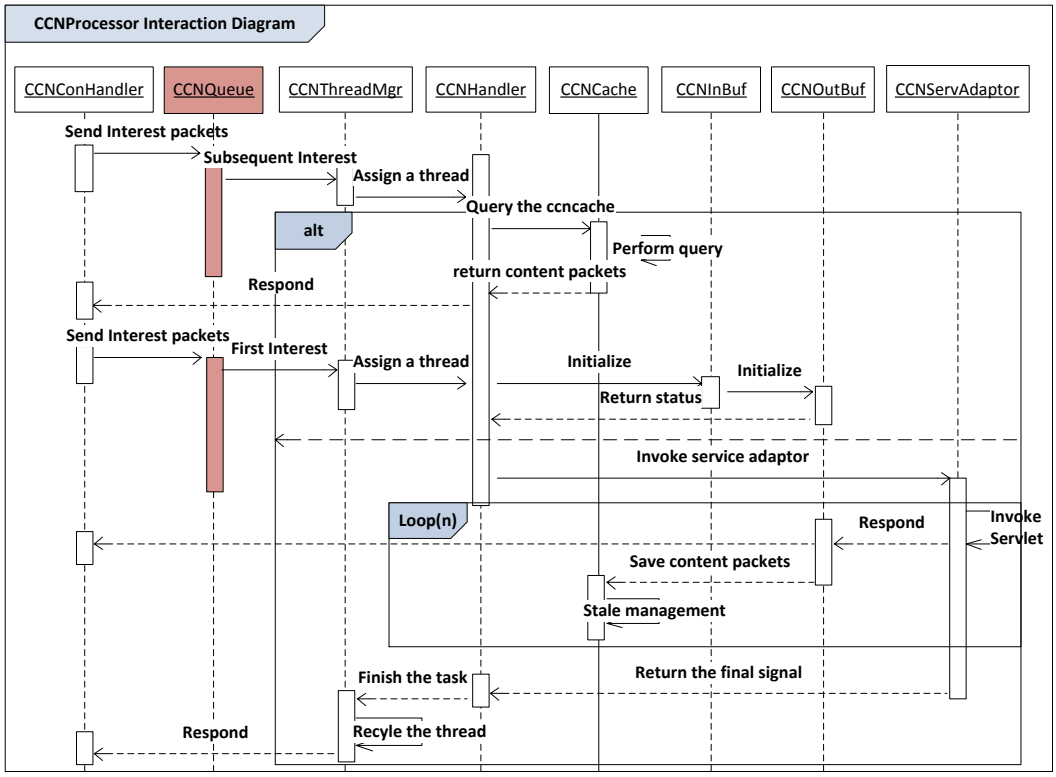
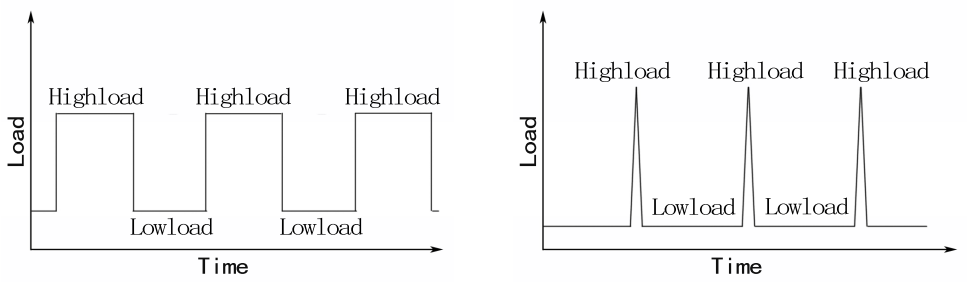


Fig. 8: Handling Process of first and subsequent Interest packets in CCNxTomcat

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

prioritizes the Interest packets of the service requests being currently processed, not the newly incoming requests. So the IDSRS approach will work when both queues are occupied by Interest packets. In general, there are two types of load, i.e. persistent load and transient load. Due to the burst of Web traffic even in the case of regular upgrade a popular Web server is still likely to experience transient periods of high load. So we consider two different types of load: In the first type, called alternating load, the load alternates between high load and low load, where the length of the high load period is equal to the length of the low load period (see Figure.9(a)); In 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.9(b)).

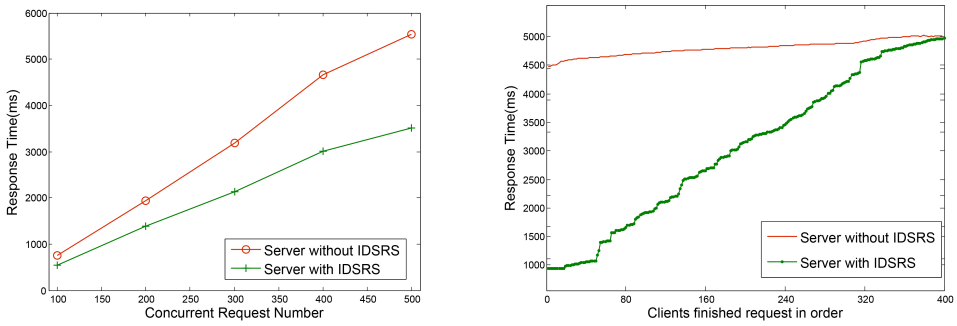


(a) Alternating load (b) Intermittent load

Fig. 9: Two different types of transient load

5.2.1. Different Response Data's Segment Numbers Distributions of Three Popular Dynamic Web Sites in China. To be closer to the reality, we investigated three popular dynamic Web sites in China, including 12306 (http://www.12306.cn/mormhweb/), the largest train ticket booking Website in China; Taobao (http://www.taobao.com), the largest C2C and B2C online shopping mall in China; Baidu (http://www.baidu.com), the most popular search engine in China. We transform their response pages' size into segment number of CCN Content packets (4KB per segment generally), and observe the distributions of their response data's segments number. These three types of dynamic Web service sites can cover most of dynamic Web types in current network. In the following part, we attempt to observe the practical impact of IDSRS on these different types of dynamic Web sites.

(1) *Case 1: Online Train Ticket Booking Website in China (12306)*: The first study target is 12306 (http://www.12306.cn), the largest online train ticket booking Website in China. It is in charge of all online querying and purchases of train tickets in China. To reduce the system's burst traffic and guarantee the fairness for customers to a certain degree, the 12306 site releases tickets hourly and most people will query and buy tickets at the same time especially before the holidays. The querying of remaining ticket information is a main service of this site, which is implemented by dynamic Web pages and the response pages' sizes are almost fixed. We simulate the user behaviors to sample some querying result pages and transform the size of each querying result page into the number of Content packets in CCN. The results show that the segment numbers approximate to Gauss distribution with the average value 15 (60KB). To test the performance, we conducted six experiments and each experiment lasts one and half hour with six peak periods. We increased the concurrent request clients from 100 to 500 in peak periods by adding 100 clients each time under the intermittent load scenario. The mean response time is used as the performance metrics. The results are shown in Fig.10
Fig.10(a) shows that server with IDSRS can lower almost half of the mean response time. We can find more details in Fig.10(b). In Fig.10(b), there are 400 clients request in one peak period, and server with IDSRS will prioritize the Interest packets of the service requests being currently processed, not the newly incoming requests. It means that the dynamic Web requests arrive at server first will be served first instead of suffering a long waiting time, which offers better service quality to some clients. For this kind of dynamic Web sites whose response data's sizes are similar and the requests are relatively intensive, the proposed IDSRS has a great performance improvement and the whole response time is reduced by 41.93%.

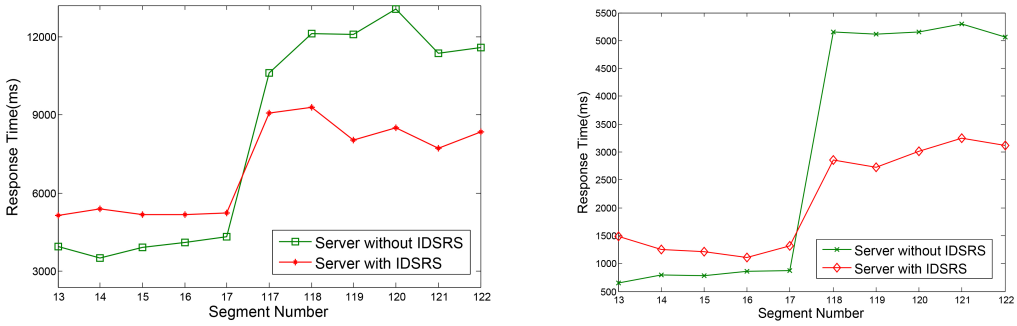


(a) Mean response time under different concurrent requests

(b) Response time under 240 concurrent requests

Fig. 10: The segment numbers of response Web pages obey Gaussian distribution and server in the intermittent load

(2) *Case 2: Online Shopping Website in China (TaoBao)*: Taobao is the largest online C2C shopping Website in China. When users shop online in Taobao Website, they often search for a certain commodity by keywords or categories. The system will display all kinds of related commodities in a display page. If a user is interested in a specific commodity, he/she will click to browse its detailed information and the Website will display it in an independent Web page. In this shopping process, since users often pay more attention to the details of some specific commodities, the service requests for specific commodities occupy the majority of this site's requests. Consequently, in our experiments, we mainly focus on two types of dynamic Web pages: one is the display page of all querying commodities and the other is the display page of specific commodities. Similar to the Case 1, we sampled a large number of response pages from Taobao, and model their sizes into CCN content segments. We find that the segment numbers of the display pages of specific commodity approximate to a Gauss distribution with average value 15, and the segment numbers of the display pages of all querying commodities approximate to a Gauss distribution with average value 120. Therefore, in this experiment, we also conducted six experiments and generated clients from 100 to 500 in peak periods by adding 100 clients each time under the intermittent load, among them, 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.11(a). Under the alternating load, we generate clients from 50 to 300. The results are shown in Fig.11(b).
Fig.11(a) shows that server with IDSRS is friendly to the dynamic Web requests with large-size response pages and increases the response time of requests with small-size response pages. In this scenario, the IDSRS approach increased the delay by 23% to small-size requests, but saved the delay by 31.85% to large-size requests. Fig.11(b) shows that under the alternating load, server with IDSRS is also more friendly to the dynamic Web requests with large-size response pages. As we can see, when the dynamic Web requests with small-size response pages occupy the majority of the site's requests, the overall average response time of the server with IDSRS has no obvious improvements compared with the server without IDSRS. Therefore, whether the server employs the IDSRS or not, it depends on the importance of these two types of dynamic Web requests. If the dynamic Web re-



(a) Under intermittent load (b) Under alternating load

Fig. 11: Mean response time of dynamic Web requests with different segment numbers' response pages, and the segment numbers of response Web pages obey the double Gaussian distributions under both loads

quests with large-size response pages play an important role in Websites, IDSRS is a good choice. Otherwise, the server without IDSRS performs better.

(3) *Case 3: Search Engine Website in China (Baidu)*: Web search is a very typical dynamic Web application in current Internet. Hence, we also investigated the performance of CCN Web server with IDSRS for this type of dynamic Web application. We selected Baidu, the largest search engine in China, as our study object. Given that the search engine uses the different keywords to generate the related content dynamically for clients, we use Baidu Index (http://index.baidu.com) to get the popularity of different keywords. We counted the search index of different keywords for a year, from March, 2014 to February, 2015. Then we related the search index to the segment numbers of search pages' size of different keywords (see Fig.12). It can be observed that the more popular the keywords, the larger the search result pages's size.

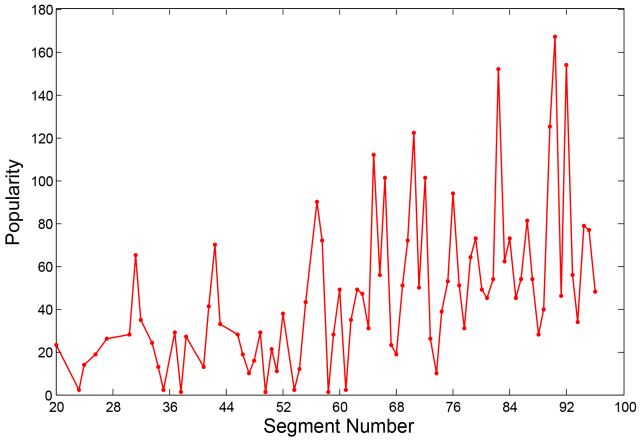
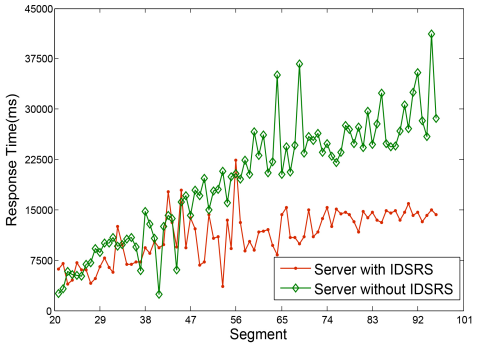
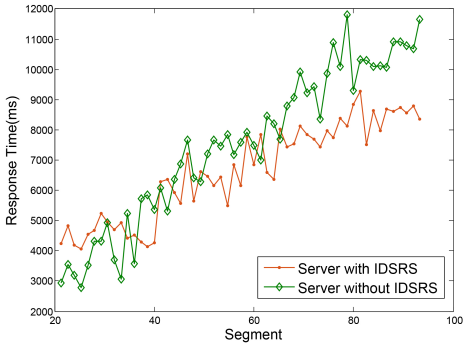


Fig. 12: The relationship of different keywords' popularity with the segment numbers of corresponding searching result pages



(a) Mean response time of different segment number under intermittent load



(b) Mean response time of different segment number under alternating load

Fig. 13: Under the intermittent load, the experiment results of Baidu search application scenario

In the experiment, we use these keywords to generate the users' searching requests. The numbers of search requests and the segment numbers of the searching result pages approximate to the distribution trend shown in Fig.12. We conducted six experiments under the intermittent load, and increased the concurrent request clients from 100 to 500 in peak periods by adding 100 clients each time. Under the alternating load, we generate clients from 50 to 300. The results are shown in Fig.13.

Fig.13 shows the same results as we sated before. the performance of search requests with large-size result pages performs well in search engine, where in general the more popular contents generate more Web pages on Internet under both intermittent load and alternating load. In this experiment, the IDSRS approach saves the mean response time by 20.2% under intermittent load and 12.84% under alternating load.

Table III: Saving response time by CCN Web server with IDSRS

Segment Numbers Distributions	**Intermittent load**	**Alternating load**
12306	41.93%	
Taobao	1.32%	9.89%
Baidu	20.3%	12.84%

5.2.2. Experiences and Suggestions Derived from the Experiments. As shown in Table.III, in general, we can see that the IDSRS approach can efficiently improve the mean response time of dynamic Web requests for CCN Web server. However, obviously, the IDSRS approach is friendly to the dynamic Web requests with large-size response Web pages, which is in accordance with the theoretical analysis results obtained in Section 4. Therefore, whether the server employs IDSRS or not, it depends on the features of dynamic Web sites. We concluded the following suggestions for reference: (1) If the dynamic Web requests with large-size response pages are more important for this Web site (i.e. the requests with large-size response pages occupy the majority or the requests with small-size response pages occupy the majority but the requests with large-size

response pages need to be prioritized.), we recommend the CCNxTomcat with IDSRS approach. If the requests with small-size response pages are more important than other requests, the origin CCNxTomcat without IDSRS is naturally friendly to the requests with small-size response pages and has a better performance. (2) Compared with the stable service request traffic, we also find that the CCNxTomcat with IDSRS performs better under the bursty service requests traffic because it can greatly reduce the mean response time.

5.3. Adaptive Pipeline-based Dynamic Service Flow Control (APDSFC) in CCN Web Server

In CCN/NDN network, the pipeline was designed to enhance the one-to-one Interest packet/Content packet exchange by sending multiple Interests at once. Based on the improved CCNxTomcat with IDSRS, we further to verify the effects of pipeline-based dynamic Web request flow control in CCN. In this paper, we present an adaptive pipeline flow control mechanism for CCN Web server. Its basic principle is that the Web clients can dynamically adjust the pipeline window size based on the size of remaining Content packets in CCNxTomcat. Specifically, since the client doesn't know the number of Content packets generated dynamically, the client will firstly send the first Interest packet to server to trigger the dynamic Web execution. After the response Content packets are generated, the server will return the first Content packet with the pipeline window size to the client. Then, the client will submit multiple subsequent Interest packets every time based on the pipeline size it received to enhance the communication efficiency (see Fig.14). We call this approach as APDSFC (Adaptive Pipeline-based Dynamic Service Flow Control).

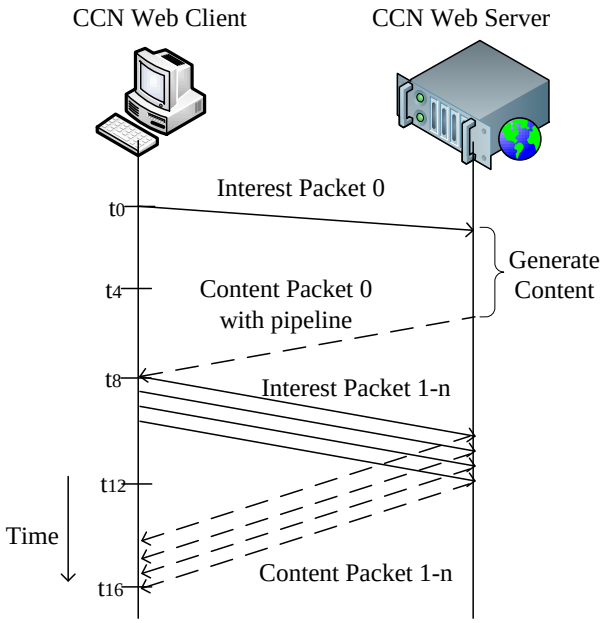


Fig. 14: Adaptive pipeline-based dynamic service flow control (APDSFC) in CCN Web server

It can be seen that the core idea of APDSFC is that the pipeline size is based on the size of the remaining Content packets. In this paper, we will test the mean response time when pipeline is $\frac{n}{q}$, where n is the number of remaining Content packets and q will vary from 10 to 1. Thus, the different dynamic Web requests will have different pipeline size based on the sizes of their corresponding response Content packets. In the following, we conducted two experiments to verify the effect of APDSFC approach.

5.3.1. Case 1: One client sends Interest packets with different pipeline sizes. In this experiment, to see the benefit of pipeline, we generate one client to request the different dynamic Web with different Content packets number from 1 to 150 with different pipeline sizes. The results in Fig.15 show that the larger the pipeline window size, the better the effect. Ideally, the fastest way is that after receiving the first Content packets with the number of remaining Content packets, the client emits all remaining Interest packets to fetch the remanning Content packets at once. In this fastest way, the APDSFC approach can reduce the mean response time by 61.9%. However, in reality, the maximum number of Interest packets sent simultaneously by client is determined by the available uplink utilization. In other words, the Interest packets themselves can create congestion causing slower request rate. This performance problem becomes more apparent in a network with asymmetric links such as cellular network [Byun et al. 2013].

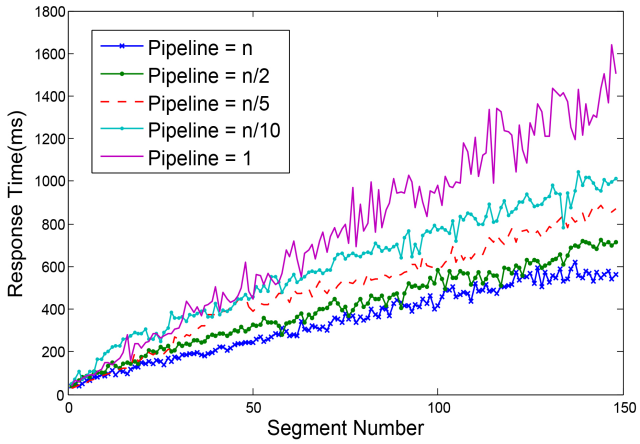


Fig. 15: Mean response time of increasing response data size under different pipeline sizes

5.3.2. Case 2: Multiple concurrent clients send Interest packets with different pipeline sizes. To further observe the effect of APDSFC approach in different distributions of response Web page sizes, we generate 60 clients, and the response Web pages' sizes obey the distributions we stated above, i.e. single Gaussian distribution (12306 Web site), double Gaussian distribution (Taobao Web site) and Baidu popularity distribution, under the intermittent load. First, we test the single Gaussian distribution scenario. The results in Fig.16 show that the APDSFC can greatly improve the response time performance of server compared with the server without APDSFC (i.e. pipeline size is 1). In this experiment, since the average value n of single Gaussian distribution is 15, the pipeline size $n/15$ is 1 (i.e. without pipeline). Increasing pipeline size results in the improvement of performance and at most it can save the response time by 45.2%.

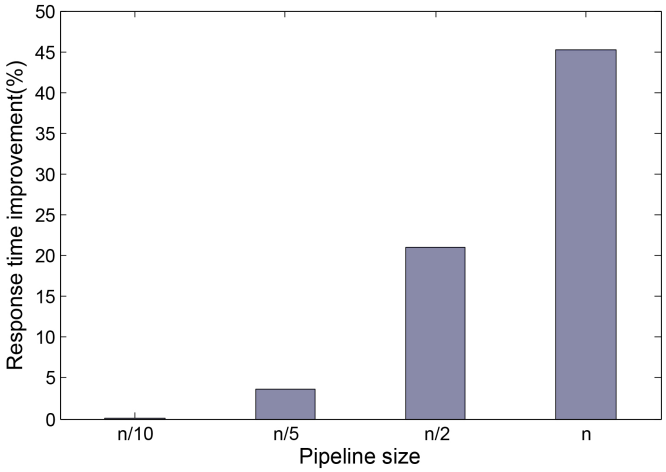


Fig. 16: Effects of APDSFC approach for response Web pages' size of single Gaussian distributions

Then we observe the effect of APDSFC approach when the response Web pages' sizes obey double Gauss distribution (i.e. Taobao Web site scenario). In Fig.17, we give the different response pages' response delays in different pipeline sizes. It can be seen that the increasing pipeline size can reduce the response delay of dynamic Web requests with large-size response contents, but increase the response delay of dynamic Web requests with small-size response contents. When the pipeline size is $n/10$ or $n/5$, there is almost no reduction in the average response time because the dynamic Web requests with small-size response Content packets occupy the majority, which brings the impact on the mean response time. When the pipeline size is n, the response time will decrease for both dynamic Web requests with small-size response contents and large-size response contents.

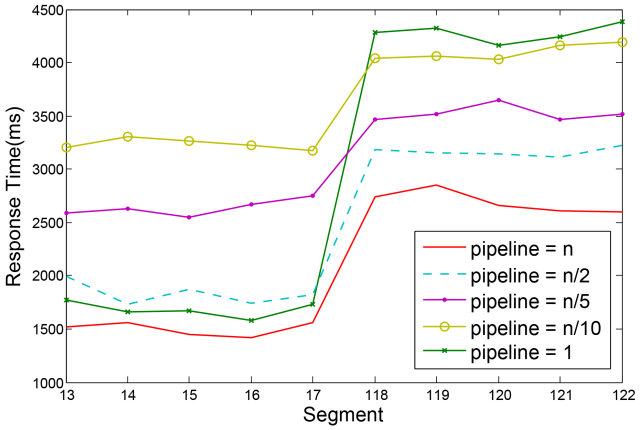


Fig. 17: Effects of APDSFC approach for response Web pages' size of double Gaussian distributions

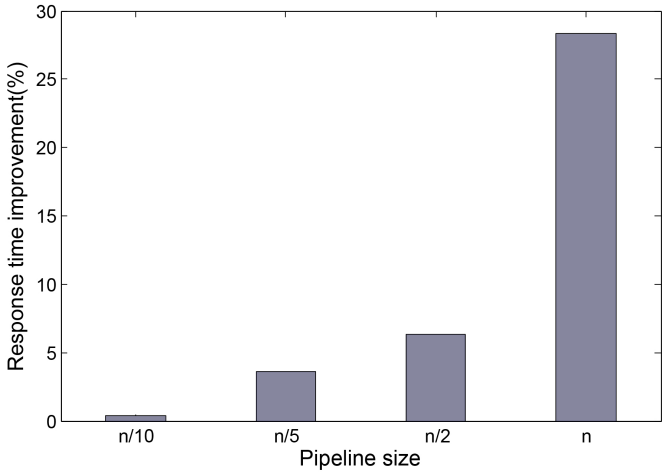


Fig. 18: Effects of APDSFC approach for response Web pages' size of Baidu popularity distribution

Finally, when it comes to Baidu popularity distribution, as shown in Fig.18, since the dynamic Web search requests with large-size contents occupy the majority, the APDSFC approach can reduce the mean response time in all pipeline sizes obviously.

6. DISCUSSIONS AND LESSONS LEARNED

In improving the dynamic Web application performance of CCN Web servers, we have gained much experience and insight in the impact of a clean-slate future Internet architecture on the upper-layer application provisioning.

Firstly, since CCN employs a completely different communication model from exiting IP network, the CCN Web servers involve in a much fine-grained scheduling, i.e. Interest packet level. However, the exiting HTTP Web servers are based on TCP connections and mainly involve in the service-level scheduling issue, i.e. different Web requests. The difference is that: in a CCN network, a dynamic Web request often consists of multiple Interest packets and each Interest packet is an independent named-based routing request. The first Interest packet is responsible for triggering the back-end script logic to generate Web content dynamically, and the subsequent Interest packets only fetch the remaining Content packets from CCN Web Servers when the response data is large than one Content packet. Obviously, this independent Interest packet-based request processing mechanism is completely different from the socket-based HTTP Web server processing mechanism. Hence, this paper sheds light for researchers to deeply understand the distinctive running mechanism of CCN Web servers.

The second lesson is related to the native communication paradigm of CCN. The existing design principles and implementation of CCN network (i.e. CCNx) are very friendly to the content distribution [Almishari et al. 2014], such as the static Web content and video on demand. Since the ubiquitous in-network caching and one Interest packet corresponding to one Content packet, the content consumer can fetch the sharable static contents from the closer routers. However, on one hand, most of the dynamic Web applications provide the personalized contents which cannot be shared by other users generally; on the other hand, since all Interest packets of a dynamic Web request must be routed to the same origin Web Servers, the existing communication pattern that one packet returns at most one Content packet is inefficient for

dynamic Web applications. The reason is that it always need take several round trips to receive multiple Content packets although the pipeline-based flow control can reduce the waiting time. Therefore, it's necessary to improve the basic communication pattern of CCN network to support the diverse upper layer application types. For example, CCN should also support a special long-time Interest packet communication pattern, which means that one Interest packet can receive multiple content packets and the subsequent Content packets can be sent back by tracing back the footprints in the PIT (Pending Interest Table) left by the first Interest packet. However, this approach will violate the basic flow balance rule in the CCN network. In addition, this approach will also bring some negative impacts on the consumer's mobility. Hence, it's still an open issue to be explored extensively for designers of network architecture and upper layer applications.

The third lesson is the scheduling issue of different kinds of services. In this paper, we only consider the scheduling of Interest packets, which guarantees that the Web server accomplishes the first-come services preferably. Although the average response time is reduced greatly on the whole, this proposed approach is unfriendly to the small-size dynamic Web requests. On the perspective of the service request level, it's still a fair First-Come-First-Served model without differentiating the small-size requests from the large-size requests. In this way, the small-size requests must wait behind the large-size requests. In fact, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling policy, allowing the small-size requests to preempt the large-size requests is desirable because forcing large-size requests to wait behind small-size requests results in much lower mean response time than the situation where small-size requests must wait behind large-size requests. In CCN Web Servers, after the process of the first Interest packet, the remaining Content packets' sizes of each request are known. Therefore, it still has further performance improving room for CCN Web Servers by prioritizing the subsequent Interest packets of small-size requests.

The fourth lesson is the real dynamic Web request behaviors. In the experiments, we investigate three mainstream dynamic Web sites in China. We use the Web crawler to simulate user behaviors to initiate the dynamic Web requests, and analyze the fetched real response data's sizes of different kinds of dynamic Web requests. In this way, the general response data's size distributions of dynamic Web requests can be acquired, which help to simulate the size of Content packets generated by CCN Web Servers. However, since there are no real user visit behaviors data, in the experiment, we can only use the CCN clients to simulate the users' visit requests according to a traffic fluctuation model. Therefore, it's better to use a trace-driven simulation when acquiring the real user visit data of these three popular dynamic Web sits in China and understand how users browse the Web [Leiva and Viv 2013].

7. CONCLUSION AND FUTURE RESEARCH

As a new clean-slate future Internet architecture, Content-Centric Networking (CCN) continues to be a favorite network research topic. To better support the upper layer Internet applications is the main motivation. However, since CCN employs a completely different communication pattern and protocol stack, the exiting application tools need to be redesigned to run over the new network. Currently, Web has become an fundamental Internet application platform and evolved from static pages to rich media applications with heavy client-side dynamic interactions, such as social networking service, content search service and cloud service. Therefore, how CCN can support modern Web communication patterns effectively is an interesting topic.

Web servers play a key role in satisfying the needs of dynamic Web requests. The existing implementation of CCN Web server (i.e. CCNxTomcat) fairly processes the Interest packets without differentiating the services being processed from the new-

ly incoming service requests, which results in a long mean user-perceived response delay. In this article, we differentiate the first Interest packets and the subsequent Interest packets based on a novel queuing discipline, and prioritize the subsequent Interest packets of services being processed. This scheduling policy guarantees that the services being processed can be accomplished preferably, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling theory. We formalize this scheduling issue and give the qualitative analysis model. In a LAN experiment network of CCN, by simulating the response data's size of three mainstream dynamic Web sites in China, the proposed IDSRS (Interest packet-based Dynamic Service Request Scheduling) scheduling algorithm reduces the mean response time significantly over the existing implementation algorithm in most cases. However, this improvement comes at cost to newly incoming dynamic Web requests with small-size response data, which have to wait behind the services being processed with large-size response data.

Based on the experience and lessons learned, in future work we plan to further improve the scheduling policy. On one hand, the mean response time is an important consideration; at the same time, the small-size dynamic Web requests are also should be prioritized. In addition, the dynamic Web applications are so important in today's Internet, and therefore it's highly desirable that they should be supported effectively in a CCN network. Since the existing CCN communication patterns are not efficient to dynamic Web interactions, we plan to explore a better CCN communication pattern without violating its basic design principles. Another important work is to collect some real user visit behavior data to conduct a complete trace-driven experiment to test the performance. At last, based on the improved CCNxTomcat, we also plan to develop more Web applications to test CCN with real-life traffic.

REFERENCES

Mishari Almishari, Paolo Gasti, Naveen Nathan, and Gene Tsudik. 2014. Optimizing Bi-Directional Low-Latency Communication in Named Data Networking. *ACM SIGCOMM Computer Communication Review* 44, 1 (2014), 13–19.

Apache Tomcat 2014. Apache Tomcat. (2014). http://tomcat.apache.org/.

Nina Bhatti and Rich Friedrich. 1999. Web server support for tiered services. *IEEE Network* 13, 5 (1999), 64–71. DOI:http://dx.doi.org/10.1109/65.793694

Ernst W. Biersack, Bianca Schroeder, and Guillaume Urvoy-Keller. 2007. Scheduling in practice. *ACM SIGMETRICS Performance Evaluation Review* 34, 4 (2007), 21–28. DOI:http://dx.doi.org/10.1145/1243401.1243407

Dojun Byun, Byoung-Joon (BJ) Lee, and Myeong-Wuk Jang. 2013. Adaptive flow control via Interest aggregation in CCN. In *Proceedings of 2013 IEEE International Conference on Communications (ICC) (ICC'2013)*. IEEE, Budapest, Hungary, 3738–3742. DOI:http://dx.doi.org/10.1109/ICC.2013.6655136

Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. 2012. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In *Proceedings of IEEE INFOCOM 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN'2012)*. IEEE, Orlando, Florida, USA., 304–309. DOI:http://dx.doi.org/10.1109/INFCOMW.2012.6193510

CCNx library 2014. CCNx library. (2014). http://www.ccnx.org/releases/ccnx-0.8.0.tar.gz.

Huamin Chen and Prasant Mohapatra. 2003. Overload control in QoS-aware web servers. *Computer Networks* 42, 1 (May 2003), 119–133. DOI:http://dx.doi.org/10.1016/S1389-1286(03)00178-6

Andrea Detti, Matteo Pomposini, Nicola Blefari-Melazzi, and Stefano Salsano. 2012. Supporting the web with an information centric network that routes by name. *Computer Networks* 56, 17 (2012), 3705–3722. DOI:http://dx.doi.org/10.1016/j.comnet.2012.08.006

Phillipa Gill, Martin Arlitt, Niklas Carlsson, Anirban Mahanti, and Carey Williamson. 2011. Characterizing Organizational Use of Web-Based Services: Methodology, Challenges, Observations, and Insights. *ACM Transactions on the Web* 5, 4 (2011), 1–23. DOI:http://dx.doi.org/10.1145/2019643.2019646

Chris Grier, Shuo Tang, and Samuel T. King. 2011. Designing and Implementing the OP and OP2 Web Browsers. *ACM Transactions on the Web* 5, 2 (May 2011), 1–35. DOI:http://dx.doi.org/10.1145/1961659.1961665

Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. 2003. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems* 21, 2 (May 2003), 207–233.

Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. 2012. Networking Named Content. *Commun. ACM* 55, 1 (Jan. 2012), 117–124. DOI:http://dx.doi.org/10.1145/2063176.2063204

Van Jacobson, D. K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca L. Braynard. 2009. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT 09)*. ACM Press, New York, NY, 1–12. DOI:http://dx.doi.org/10.1145/1658939.1658941

Luis A. Leiva and Roberto Viv. 2013. Web Browsing Behavior Analysis and Interactive Hypervideo. *ACM Transactions on the Web* 7, 4 (2013), 1–28. DOI:http://dx.doi.org/10.1145/2529995.2529996

Chenyang Lu, Tarek F. Abdelzaber, John A. Stankovic, and Sang Hyuk Son. 2001. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the seventh IEEE Real-Time Technology and Applications Symposium (RTAS'2001)*. IEEE, 51–62.

David T McWherter, Bianca Schroeder, Anastassia Ailamaki, and Mor Harchol-Balter. 2005. Improving preemptive prioritization via statistical characterization of OLTP locking. In *Proceedings of the 21st International Conference on Data Engineering (ICDE' 2005)*. IEEE, 446–457.

Siddharth Mitra, Mayank Agrawal, Amit Yadav, Niklas Carlsson, Derek Eager, and Anirban Mahanti. 2011. Characterizing Web-Based Video Sharing Workloads. *ACM Transactions on the Web* 5, 2 (May 2011), 1–27. DOI:http://dx.doi.org/10.1145/1961659.1961662

Ilya Moiseenko, Mark Stapp, and David Oran. 2014. Communication patterns for web interaction in named data networking. In *Proceedings of the 1st international conference on Information-centric networking (ICN'2014)*. ACM, Paris, France., 87–96. DOI:http://dx.doi.org/10.1145/2660129.2660152

Guoshun Nan, Xiuquan Qiao, Yukai Tu, Wei Tan, Lei Guo, and Junliang Chen. 2015. Design and Implementation: the Native Web Browser and Server for Content-Centric Networking. In *Proceedings of the ACM SIGCOMM Poster (SIGCOMM15)*. ACM, London, United Kingdom. DOI:http://dx.doi.org/10.1145/2785956.2790024

Diego Perino and Matteo Varvello. 2011. A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking (ICN' 11)*. ACM, 44–49. DOI:http://dx.doi.org/10.1145/2018584.2018596

Xiuquan Qiao, Guoshun Nan, Yue Peng, Lei Guo, Jingwen Chen, Yunlei Sun, and Junliang Chen. 2015. NDNBrowser: An extended web browser for named data networking. *Journal of Network and Computer Applications* 50 (April 2015), 134–147. DOI:http://dx.doi.org/10.1016/j.jnca.2014.06.009

Xiuquan Qiao, Guoshun Nan, Wei Tan, Lei Guo, Junliang Chen, Wei Quan, and Yukai Tu. 2014. CCNx-Tomcat: An extended web server for Content-Centric Networking. *Computer Networks* 75, Part A (Dec. 2014), 276–296. DOI:http://dx.doi.org/10.1016/j.comnet.2014.10.014

Natalya Rozhnova and Serge Fdida. 2012. An effective hop-by-hop Interest shaping mechanism for CCN communications. In *Proceedings of IEEE INFOCOM 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN'2012)*. IEEE, Orlando, Florida, USA., 322–327. DOI:http://dx.doi.org/10.1109/INFCOMW.2012.6193514

Linus E. Schrage and Louis W. Miller. 1996. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research* 14, 4 (1996), 670–684. DOI:http://dx.doi.org/10.1287/opre.14.4.670

Bianca Schroeder and Mor Harchol-Balter. 2006. Web Servers Under Overload: How Scheduling Can Help. *ACM Transactions on Internet Technology* 6, 1 (2006), 20–52.

Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, and Erich Nahum. 2006a. Achieving class-based QoS for transactional workloads. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE' 2006)*. IEEE, 153–155. DOI:http://dx.doi.org/10.1109/ICDE.2006.11

Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, Erich Nahum, and Adam Wierman. 2006b. How to determine a good multi-programming level for external scheduling. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE' 2006)*. IEEE, 60–71. DOI:http://dx.doi.org/10.1109/ICDE.2006.78

Wentao Shang, Jeff Thompson, Jeff Burke, and Lixia Zhang. 2013a. *Development and Experimentation with NDN-JS, a JavaScript Library for Named Data Networking*. Technical Report NDN-0014. Department of Computer Science, Center for Research in Engineering, Media and Performance, UCLA.

Wentao Shang, Jeff Thompsonz, Meki Cherkaoui, Jeff Burkey, and Lixia Zhang. 2013b. NDN.JS: A JavaScript Client Library for Named Data Networking. In *Proceedings of 2013 IEEE Computer Communications Workshops (INFOCOM WKSHPS'2013)*. IEEE, 399–404. DOI:http://dx.doi.org/10.1109/INFCOMW.2013.6970726

Sergio Duarte Torres, Ingmar Weber, and Djoerd Hiemstra. 2014. Analysis of Search and Browsing Behavior of Young Users on the Web. *ACM Transactions on the Web* 8, 2 (2014), 1–54. DOI:http://dx.doi.org/10.1145/2555595

Sen Wang, Jun Bi, Jianping Wu, and Xu Yang. 2012a. On adapting http protocol to content centric networking. In *Proceedings of the 7th International Conference on Future Internet Technologies (CFI'2012)*. ACM, 1–6.

Sen Wang, Jianping Wu, and Jun Bi. 2012b. Application Design over Named Data Networking with Its Features in Mind. In *Proceedings of The Eleventh International Conference on Networks (ICN'2012)*. IEEE, 121–124.

Yi Wei and M. Brian Blake. 2014. Proactive Virtualized Resource Management and Provisioning for Service Workflows in the Cloud. *Computing* (2014). DOI:http://dx.doi.org/10.1007/s00607-014-0419-4

Nong Ye, Esma S Gel, Xueping Li, Toni Farley, and Ying-Cheng Lai. 2005. Web server QoS models: applying scheduling rules from production planning. *Computers & Operations Research* 32, 5 (May 2005), 1147–1164.

Guohua You and Ying Zhao. 2012. A weighted-fair-queuing (WFQ)-based dynamic request scheduling approach in a multi-core system. *Future Generation Computer Systems* 28, 7 (2012), 1110–1120. DOI:http://dx.doi.org/10.1016/j.future.2011.07.006

Haowei Yuan and Patrick Crowley. 2013. Experimental Evaluation of Content Distribution with NDN and HTTP. In *Proceedings of the IEEE INFOCOM (INFOCOM13)*. IEEE, Turin, Italy, 240–244. DOI:http://dx.doi.org/10.1109/INFCOM.2013.6566771

Haowei Yuan, Tian Song, and P. Crowley. 2012. Scalable NDN forwarding: Concepts, issues and principles. In *Proceedings of 2012 21st IEEE International Conference on Computer Communications and Networks (ICCCN 2012)*. IEEE, 1–9. DOI:http://dx.doi.org/10.1109/ICCCN.2012.6289305

Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named Data Networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73. DOI:http://dx.doi.org/10.1145/2656877.2656887