I.J. Computer Network and Information Security, **2014, 7, 1-10**
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2014.07.01

Router-based Content-aware Data Redirection for Future CDN Systems

Janaka L. Wijekoon
Hiroaki Nishi Laboratory, Department of Computer Science, Keio University Hiyoshi 3-14-1, Yokohama, Japan
Email: janaka@west.sd.keio.ac.jp

Erwin H. Harahap, Shinichi Ishida, Rajitha L. Tennekoon and Hiroaki Nishi
Hiroaki Nishi Laboratory, Department of Computer Science, Keio University Hiyoshi 3-14-1, Yokohama, Japan
Email: {erwin2h, sin, rajitha}@west.sd.keio.ac.jp and west@sd.keio.ac.jp

***Abstract*—**Delivery of data-enriched applications has become a top priority on the Internet, and Internet users are demanding faster and higher-quality services. Cater such requirements, Content Delivery Networks (CDNs) were introduced. However, the growth rate of information on the Internet requires infrastructural modifications to keep the consistency while maintaining quality of the Internet services. To this end, the Service-oriented Router is introduced to provide content based services by shifting the current Internet infrastructure to information-based open innovation platform. In this study, initially we provide implementation notes of a software-designed SoR. Then we propose a new method of CDN Request Redirection (RR) (SoR-based RR), which is designed to redirect packets based on the content of packets and the status of content servers using an SoR as an edge router of a CDN. Furthermore, we present the design and implementation of a prototype to realize the SoR-based RR in a testing network. By analyzing the result of the prototype implementation, we show that the SoR-based RR can enhance the both client experience and faster adaptations to the server changes in CDN environments.

***Index Terms*—**Service-oriented Router, Network Simulation, Data Analysis, Content Delivery Networking, Request Redirection.

I. Introduction

Communication technologies have become significantly advanced in the past few decades, with new technologies being invented to achieve efficient communication. In addition, people are more interested in sharing knowledge and information for various purposes. Innovations in information sharing are continuously accelerated to cater user needs in such environments. These motives have encouraged the construction of sophisticated environments for effective communication and information delivery.

Content Delivery Networks (CDNs) are implemented to achieve low-latency content delivery such as; data streaming, on-line gaming and e-commerce web accesses by placing content servers near the customer [1]. RR techniques are used in CDNs to redirect client requests to the nearest surrogate server [1-7]. Therefore, it is important to maintain a better RR method that can find the nearest server for a particular user. Besides, a RR method can also achieve efficient server load balancing [6, 10]. DNS redirection and URL rewriting [1] are the most commonly used CDN redirection methods in the current CDNs.

Several studies have been conducted on the impact of content delivery on networks in the past few decades. Before the adoption of worldwide server caching, such as CDN, some early research on the effectiveness of interior web caching were performed by Gadde et al [8]. In addition, recent studies [9, 10] indicate the DNS-based method can significantly reduce the download latency. Top-level CDN providers, such as Akamai [3, 4] and CenturyLink [5], use DNS-based RR to redirect client requests to the nearest surrogate server, or sometimes known as a redirector or an edge server. However, all those studies use multi tire name servers in order to maintain the consistency of changes of surrogate servers [3, 6, 11].

The DNS-based redirection use small Time to Live (TTL) values at the clients in order to adapt the changes occurs on the CDN infrastructure [1, 3, 4]. Consequently, small TTL values lead the client to access the local name server frequently to get an up to date infrastructural changes [1, 11 - 13]. Furthermore, name server resolving process increases the client waiting time due to communication between local name server and authoritative name servers to find the CDN surrogate servers' IP addresses. Poese et al. [6] proposed a content aware traffic engineering design that give some hints about the ISP collaborated method of redirecting packets based on both status of the servers and the network. But that study also was not able to provide a solution to the clients regarding the latency factors of connection initiation and lags of the connection due to the DNS resolving process and small TTL values. Therefore, we argue that the usage of multi tire name servers and small Time to Live (TTL) values on DNS-based RR considerably degrades the connection initiation process.

We argue that, if an edge router of the network can identify the server locations and redirects client request to the servers regardless of the IP address of the final

destination, we can utilize data redirection process to achieve faster connection initiation and faster adaptation to the server changes in a CDN. To this end, this study proposes a redirection mechanism which is redirecting

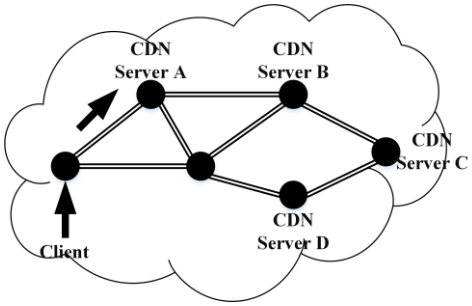


Fig. 1. SoR-based RR Overview

packets based on the content of the packet by using an Service-oriented Router [14, 15, 16] as an edge router of a network. Furthermore, we confer that the basic functions of the SoR such as; DPI and content based data redirection, introduce the ideal test bed to implement a prototype for an edge router based date redirection. Fig. 1 illustrates the basic concept of SoR-based RR. The client sends the data stream to the SoR which is placed at the edge of network and SoR selects the proper surrogate; A, B, C or D, based on the content of the packet and the status of the server and, redirects the packet stream to the selected server. In addition, an SoR redirects packets by performing server load balancing if an SoR has routing records of more than one surrogate server.

The paper is arranged as follow. A review of the DNS-based RR is provided in the Section 2. Section 3 describes the motivation of this study. Section 4 describes the software implementation of the SoR followed by the detailed explanation of prototype implementation of the SoR-based RR in Section 5. Test cases designs, implementations and the test results are presented in the Section 6. Section 7 is used to discuss the effectiveness of the SoR for the CDN redirection process and how the SoR-based RR is able to overcome the limitations of existing methods. Section 8 concludes the paper with tentative future implementations.

II. BACKGROUND STUDY

The Domain Name System (DNS) is a distributed database of records (name-to-address mapping) cached at intermediate name servers [17]. Each record has a time-to-live (TTL) value that indicates how long it will be cached. Generally TTL values are set to one day [17]. The original operation of the DNS is to resolve a Fully Qualified Domain Name (FQDN) to its corresponding IP address [11, 17].

However, in CDN infrastructures, once the DNS is used to resolve the nearest surrogate server's IP for a particular end-user, the DNS cache becomes a significant obstacle to providing real-time control of the CDN. The name resolution cache must be disabled by setting TTL values to zero in order to obtain real-time control in CDNs [1]. Small TTL values allow fine-grained load balancing and rapid response to changes in the server or network load. Nonetheless, small TTL values force clients to contact the authoritative name server for every name resolution request. However, this phenomenon increases access latency for particular content.

To understand the behavior of the DNS in a real network, we captured and analyzed the traffic of a major Internet backbone network for academics in Japan. The most noticeable fact was that the occupation ratio of the DNS, which was occupied more than 25% of traffic during the captured time. Though we cannot conclude that the measured DNS traffic occupation was entirely from CDN networks, this clearly highlights that the DNS traffic is high in the Internet data exchange.

On the other hand, recent studies [6, 18, 19] have been done to research about the ISP involvement of the DNS redirection. Authors have theorized that the ISP-CDN and DNS collaboration make the ideal network infrastructure for faster data delivery on the Internet. Those studies suggested of placing low level name servers at the ISP. Then redirect clients to the nearest server based on the status of both ISP's network the status of the content servers. However, this method also requires the clients to frequently contact local name server in order to get the status of the CDN upon the expiration of TTL values [1, 13, 19]. Besides, Anees et al. [1] confirmed that the DNS-based server redirection must be based on the name server's location, not on the client's location. Therefore, the response delay associated with connection initiations and request redirections depends on the number of hops to the authoritative name server. That study pointed out that aforementioned factors are significant factors in data redirection and client response time.

III. MOTIVATION

Literature related to the DSN-based RR [1, 20-22] shows that the DNS-based RR method has following limitations while selecting and redirecting client connections to a particular content server. First, DNS has to travel through many tires of name servers to identify the nearest content server for a particular client [3, 4]. Second, users are required to find the nearest surrogate server's IP address before initiating a connection with the content server [6, 7, 10]. As a result, time taken to initiate a connection is comparatively high. Furthermore, due to the low TTL values, clients are required to query local name servers in short time intervals in order to adapt to the network changes. Besides, multi tires of name server resolving create lags in the middle of data transmission. As an example, lags in an online game playing due to sudden server changes, and lags in video streaming will degrade the user experiences. It will be critical for customer satisfaction.

To this extend, as shown in the Fig. 1, in this paper, we propose a method to redirect packets based on the content from an edge router of the network. As a matter of fact, a network router connects several independent networks and forwards data from a source to a destination. Yet

general routers are not capable of providing content-based services, and this limitation restricts their benefits for both users and network carriers.

As a test-bed of a router providing content-based services, service-oriented router (SoR) [14, 15, 23] was proposed to enable data collection by stream recovery and deep packet inspection (DPI) and to provide content

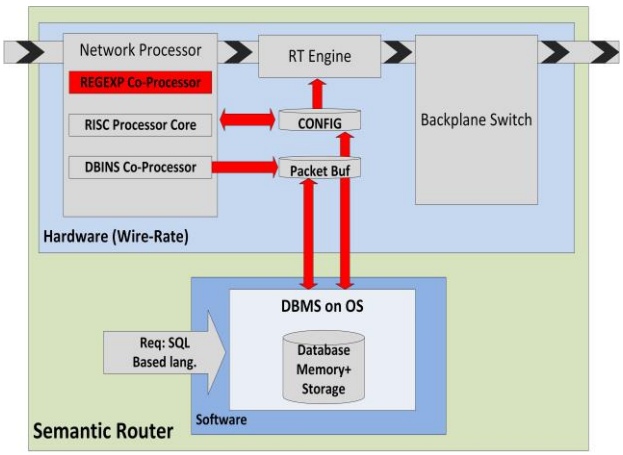


Fig. 2. Semantic Router Infrastructure Discussed in [15, 23]

based services from the router itself. The SoR; as shown in Fig. 2, is specially designed router for providing services to end users from the router itself by using a dedicated APIs, which was proposed and implemented by our research team [16]. As explained in [15, 23], packet stream analyzing on the router and providing a contents-based services based on the analysis results proofs the effectiveness of the SoR as a service providing device.

In this study, as illustrated in Fig. 1, we placed an SoR as an edge router of the network and the SoR was programmed to redirect packets: 1.) based on the content of the packet, and 2.) based on the status of the content servers. Moreover, SoR communication protocol was enhanced to advertise server status such as queue length to the edge routers (SoRs). Therefore, SoRs are able to detect the locations of the content servers and broadcast servers' locations among the SoRs in the network. In addition, SoRs redirect the data packets to the nearest possible content server upon a packet arrival without depending on the destination IP address. Consequently, the SoR-based RR method does not require client to have exact destination IP address to initiate a connection. SoR-based RR method requires the client to send data to the edge SoR and the edge SoR redirects the packet to a suitable content server. That flexibility enables the client to send packet streams without waiting DNS resolve to find the surrogate server's IP address.

IV. SoR and its Underlying Technology

A. Evaluation of SoR

An SoR is a router that captures data streams passively and stores data in high-speed databases [15, 23]. An SoR is designed to capture data streams at wire rates such as 10 Gbps in effective speed [24], after which, it reconstructs the data streams and performs DPI, including Layer 7 information. That mechanism was designed to provide an API for users to design interactive applications in order to support future Internet technologies. As shown in Fig. 2, Inoue et al. proposed the Semantic Router (SR) as the initial idea of the SoR in [15] and Nagatomi et al. implemented the cache based process engine to the proposed semantic router in [23].

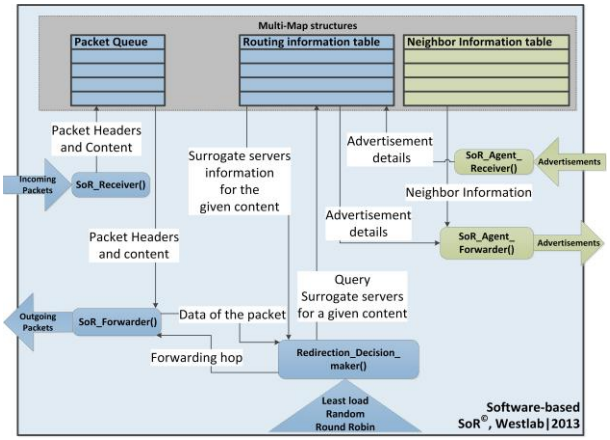


Fig. 3. Services-oriented Router Infrastructure

In contrast, the crucial problem of SoR is to maintain data throughput while performing packet analysis of Layer 7 information, storing data in databases, and retrieving data. These processes slow down the packet-forwarding process and ultimately the performance of the entire network. To address such problems, we propose hardware-level acceleration to reduce memory accesses through hardware-based accelerations [23, 25, 26]. The software-based SoR was implemented on a JunOS V app engine for the commercial Juniper router, proving that SoR can perform high-throughput processing over gigabit networks [24]. Furthermore, the ns-2 SoR module was implemented in [27] to simulate basic functions of the SoR.

However, the complete model of the SoR given in Fig. 2 is still under research and development. Consequently, in this study, we implemented a software-based SoR as given in Fig. 3. Then, we used the implemented software-based SoR to create a testing environment that we then used to simulate an SoR-based RR. The proposed software-designed SoR supports essential functions, such as 1) DPI to analyze packet information from IP layers to application layer, 2) packet-forwarding according to the content of the packet, and 3) communications between SoRs and CDN servers. The implemented software-based SoR has two major modules, packet analysis and routing module and radar module.

B. Packet Analysis and Routing Module

According to the Fig. 3, the packet analysis and routing module consists of the following sub-modules: 1) packet receiver, 2) packet forwarder, 3) packet buffer, and 4)

redirection decision maker. The packet receiver is a UNIX UDP socket that listens to dedicated port 52,000. When a packet arrives at the port, the packet receiver moves the packet to a First-In-First-Out (FIFO) buffer. Then, the packet-forwarding module takes the packet from the buffer and forwards it to the redirection decision maker module. The redirection decision maker module consists of three parts: 1) analyzer, 2) forwarding information base, and 3) load-balancer. Once the packet arrives at the analyzer module, it inspects the packet in detail, including its UDP and IP header information along with the Layer 7 data. Then the analyzer module returns

Content <hash value>	Next Hop IP Address	Content Server IP Address	Content Server Queue Length

Fig. 4. SoR Forwarding Information Base (FIB)

Packet _Type : <SoR Periodic Advertisement, Immediate Advertisement, Server Advertisement>
Cost_of_the_server
Packet_Generated_Time
SoR/Server ID <IP address>
Router/Server Type <SoR,S:server,M:server>
Data

Fig.. 5. Advertisement Packet Structures

the packet payload to the redirection decision maker module. Next, the redirection decision maker module generates a hash value using the packet payload. Then, the hash value correlates with entries in the forwarding information base given in Fig. 4, which is designed as an in-memory database. That table is used to find the corresponding next hop according to the payload of a particular packet. In that process, the redirection decision maker module finds the next hop IP addresses from the forwarding information base, using the generated hash value. The IP address of the next hop can be either SoRs or a content server. After the redirection decision maker module receives the forwarding IP addresses, the load balancing module selects one of the forwarding IP addresses based on the load balancing algorithm and returns it to the forwarding module in order to forward the packet to the next hop.

We introduced three load-balancing algorithms to the SoR, namely, Random (Rand), Round Robin (RR) and Least Load (LL). For experimental purposes, we used server queue length as the criterion of selection for the LL algorithm.

Finally, the packet forwarding module forwards the packet to the corresponding IP address returned from the packet analyzer module. This scenario is repeated in every SoR until the packet reaches its destination. Therefore, the SoR-based CDN system was designed by using a hop-by-hop data forwarding process. The hop-by-hop design allows the SoR-based CDN infrastructure to have total control over packet forwarding between the client and the server. Unlike forwarding packets based on final destination's IP address, the exact destination is not required in this routing paradigm. This means that the routing of packets can be changed dynamically at any SoR, depending on the packet's contents.

C. Radar Module

The radar module is designed to gather neighbor information, which can then be used to make a decision on hop-by-hop forwarding. Once we add the neighbor IP addresses to the SoR in the router configuration state, the SoR automatically generates neighbor information table

Packet_ID
Packet_Genarated_Time
Source_IP
Source_PORT
Packet_Modification_Indicator
Contenet_ID
Contenet_TYPE<Request,Response,Update>
Data

Fig. 6. SoR Data Packet Structure

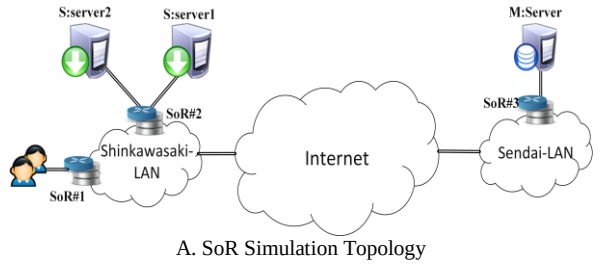
as depicted in Fig. 3. The module is capable of sending two types of advertisements: periodic advertisements and immediate advertisements. The packet structure used for the advertisements is given in Fig. 5.

When a new surrogate server is added to the network, the surrogate server contacts its affiliated SoR and transmits several pieces of information, its presence, the type of data the surrogate server supports, and the queue length of the surrogate server using the packet structure given in Fig. 5. Once an advertisement is received at the radar module, it checks for the piggybacked information for the advertisement. If the advertisement is piggybacked, the SoR omits that advertisement. If not, it generates a hash value of the content and checks that hash value with the existing hash value records in the forwarding information base. If there are no records, it adds an entry to the routing table and, sends an immediate advertisement to its neighbors about the surrogate server via a UDP socket. On the other hand, if the content server is listed in the forwarding information base, the SoR checks for the queue length and updates the particular record accordingly, and then sends an immediate update to its neighbors. Nonetheless, if any server does not contact its edge router, the particular router deletes the record from its forwarding information base and sends an immediate update to its neighbors. Unless the aforementioned scenarios have occurred, SoRs are programmed to send periodic updates. For this experiment, we assumed that 1) all SoRs are up and running, and 2) no link failure occurs, and we set the periodic update advertisement between SoRs to 10 s.

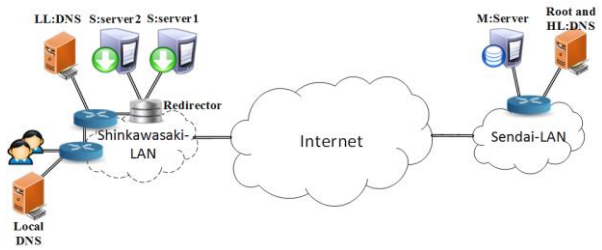
V. PROTOTYPE IMPLEMENTATION

The design of the simulation environment was based on three main modules; a traffic generator to serve as a client module, a server module, and a name server emulation module to emulate DNS. We implemented all clients, SoRs, and server programs using basic UNIX libraries such as pthread library, c++ map, c++ multi-map, c++ queue structures, and UNIX socket communication library [28].

For the experimental purposes, the SoR-based RR prototype was implemented based on UDP because, 1) UDP is a connectionless protocol and appropriate for route data in hop-by-hop routing protocol which was explained in Section 4; 2.) UDP is used in multimedia and online gaming applications [30, 31].



A. SoR Simulation Topology



B. DNS Simulation Topology

Fig 7 Simulation Topologies

A. Client Module

We designed the client as a constant UDP traffic generator which is similar to a multimedia application or online gaming application. The client module was implemented to fill a packet with the information given in Fig. 6 and send the packet to the edge router. In the client configuring state, the client was programmed with the IP address of the edge router which is always an SoR. Besides, As explained in the Section IV, the client sends the data to the edge router without resolving the IP address of the final destination. This aspect allows the client to send data without waiting for name server resolution. Then, the particular edge router (SoR) forwards the data to the surrogate server as explained in Section IV.

B. Server Module

The server application accepts user requests on the socket and stores packets in a buffer implemented as an FIFO queue. The server fetches a packet from the buffer and generates a response packet using the same packet structure given in Fig. 6, and sends it to the associated edge router.

In addition, server is configured to communicate with the edge SoR. The advertisement packet contains information such as the current queue length and IP address of the server. Once a new content has been added to the server, it immediately notifies the associated SoR about the content and the queue value. Otherwise, it sends keep-alive messages periodically to the edge SoR to inform that the server is alive and inform the present queue length. For experimental purposes, we programmed the keep-alive message interval as 5s.

C. Name Server Module

For our experiment, we created a simple socket communication program to emulate name servers. Since UDP is used in the domain name resolution [17], we created a simple c++ map to store FQDN and its associated IP address as the name server. Once a DNS query arrives at the name server, it checks the map file and returns the resultant IP address for the FQDN.

VI. SIMULATION, EXPERIMENT AND TEST RESULTS

We used a test bed with network virtualization technologies and VLAN technology to simulate a network. This environment was created by Hitachi Central Research Laboratory and presented at the Global Inter-Cloud Technology Forum (GICTF) [29]. This network has data centers at the Shin-Kawasaki campus of Keio University and Sendai, Japan. As shown in Fig. 7-A, we implemented the SoR-based CDN infrastructure on this network environment. The computational resources of the test environment were limited, but enough to implement the SoR-based RR prototype and evaluate with the DNS-based RR. We used three SoRs, one main server, two surrogate servers, and one client. SoR#1, SoR#2, and SoR#3 were used as the edge SoRs for the client, the surrogate servers, and the main server, respectively. All SoRs and servers were able to advertise themselves to their neighbors. We placed the surrogate server at the Shin-Kawasaki data center, because the surrogate server must be proximal to the client.

The simulator was designed as a packet-based simulator. Therefore, the testing scenarios are created based on the number of packet sent by the client. We measured the Round Trip Time (RTT) and the packet arrival delay in each packet returned to the client, and we used those time values as the data to evaluate the SoR-based RR method compared to the DNS-based RR methods.

As shown in the Fig. 7-B, we placed name servers similarly to their placement in Akamai's method of finding the nearest redirector [3, 4]. In this experiment, three layers of name servers; root, high-level, and low-level name servers, were used to simulate the DNS-based CDN redirection. According to [1, 6, 11], the client TTL

value is always 1 s or 2 s. However, in our study, as we use packet based simulator, we were unable to set TTL values as 1s or 2s. Therefore, we programmed the DNS-based simulation to resolve the DNS for every new content request and every change occur in servers.

Testing scenarios were created as follows. Since our main focus is to find the connection initiation delays and reveal the performance of content server interruptions in the midst of the communication in both the proposed SoR and DNS-based RR methods, we planned the following scenarios. We set the client to send 500 packets initially. Only the main server was started at the beginning of the simulation. Therefore, all packets were redirected to the main server. Then, at the time client sent the 500th packet, surrogate server was started. Then, after the client sent the 1000th packet, we temporally suspended the surrogate server. Repeatedly, the servers; main and surrogate servers, were toggled for each 500 packets. In each scenario, we measured the RTT of response packets and the latency between two response packets.

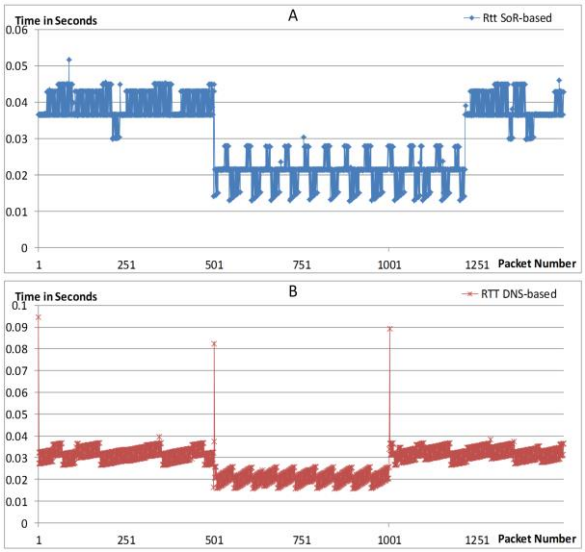


Fig. 8. RTT Comparisons

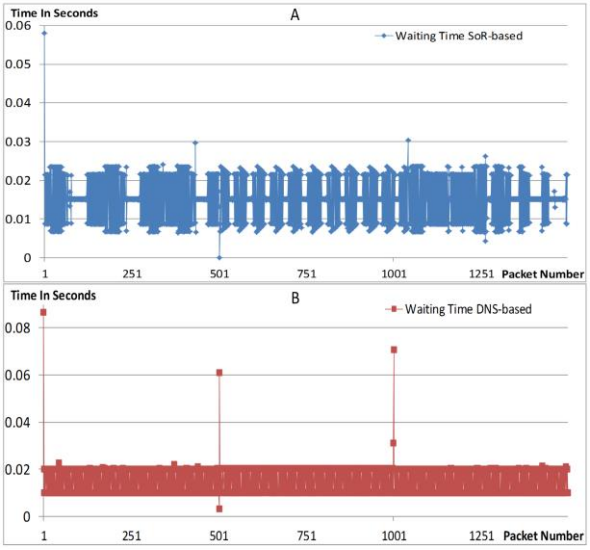


Fig. 9. Packet Inter-arrival time Comparisons

In the SoR-based method, as only the main server was started initially, the main server was able to advertise the information about the content that the server supports to SoR#3. As a result, SoR#3 was able to advertise about the main server with the other SoRs in the topology. Consequently, once the client started and sent data to the edge router which is the SoR#1, SoR#1 redirect the packets to the main server immediate after receiving. In fact, in the DNS-based method, the client was programmed to query the local name server to get the IP address of the redirector. Once the name server returned the IP address of the redirector, the client was started to send data to the main server. In both methods, we measured the RTT and the packet inter-arrival time (latency between two response packets) of the response packets and the results were plotted in the graphs shown in Figs. 8 and 9.

As shown in Fig. 8-B, for the connection initiation, the DNS-based method consumed about 0.095s. In fact, the SoR-based method was able to initiate the connection by consuming only 0.0375s, which is about 60% faster compared to the DNS based method. Furthermore, the DNS-based method shows high variances of RTT while temporary server shifting took place at the 500th, 1,000th, and 1,500th packets, owing to the required DNS resolution time. In contrast, as shown in the Fig. 8-A, the SoR-based method did not exhibit such high RTT variances. As an example, at the 500th packet, the SoR-based method consumed only 0.0125s to redirect the packet stream from the main server to the surrogate server. However, the DNS-based method consumed about 0.081s to redirect the connection from the main server to the surrogate server. In an average, the SoR-based method adopt to the server changes 58% faster compared to the DNS-based RR method.

Fig. 9-B shows that the packet inter-arrival time for the DNS-based method increases during the period of server toggling, because the DNS-based method searches for a new server while content servers were being toggled. In fact in the SoR-based method, according to the Fig. 9-A, the packet inter-arrival time does not exhibit such high increases when the servers are toggled. Packet inter-arrival time is varying between 0.009s and 0.025s for all reply packets from both servers, with no sudden latencies in the packet arrival. That phenomenon allows the client to receive data smoothly without lags in the receiving data streams.

However, based on Figs. 8 and 9, we understood that the RTT of the software-based SoR is slightly higher than a conventional backbone router, because the SoR performs DPI to determine the next hop. According to the Fig. 8, there is 0.01s RTT latency in the SoR packet forwarding. Reason is, regular routers were not programmed to DPI the packets to forward. They used simple static routing table to route the packets. Therefore, the results imply that the SoR processing delay also affects the client RTT. However, the difference in processing delay is insignificant compared to the faster

adaptation to the server changes and the faster connection initiation. Therefore, the merits of SoR are greater than its demerits in terms of communication delay. On the other hand, the RTT variation and packet inter-arrival time shifts up and down based on one baseline. As an example, in Fig. 8-A, the baseline RTT is about 0.022s for packets 501 to 1,200, because the SoR was designed entirely as software, and basic thread communication methods was used to share the single packet queue between receiving and sending functions. Therefore, the thread lock of the packet queue between sending and receiving functions resulted in such variance in the RTT. Moreover, the network latency also affected the RTT. As an example, in Fig. 8-A, packets one to 500 show RTT values between 0.0375s and 0.045s approximately, because those packets have to go through the public network in order to reach the main server at Sendai data centre.

In contrast, the RTT of packets 501 to 1,200 varied from 0.015s to 0.0275s with a baseline of approximately 0.022s, because those packets were

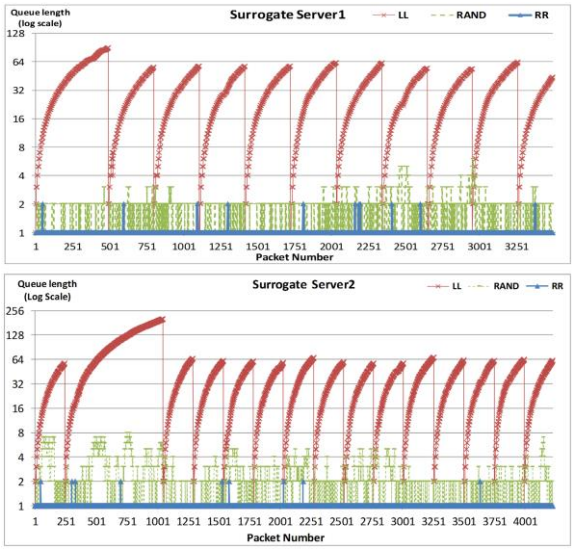


Fig. 10. Queue Length of S:Servers for Load Balancing Algorithms

destined for the surrogate server in the Shin-Kawasaki data centre together with the client. Those packets were not traveling in the public network, so the RTT is low compared to that of the packets traveling to the Sendai data centre.

As Load balancing occurs only within an ISP network or a geographically near cluster [1, 11], to compare the effectiveness of load balancing implemented for the SoR-based RR, we used only two servers attached to the Shin-Kawasaki data centre. Originally, both surrogate servers were started and let the information propagate among the SoRs in the topology. Then, we started the client and let SoR#1 and SoR#2 redirect the packets among the surrogate servers by measuring the queue values of the surrogate servers. Afterward, we measured the RTT and the packet inter-arrival time of the response packets and plotted the results in Figs. 10 and 11. As Fig. 11 illustrates, the queue lengths of both servers were low when using either the RR algorithm or the RAND algorithm, because every packet was delivered to one or the other server regardless of the server queue length. As a result, as depicted in Fig. 10, the loads of both servers were properly balanced. However, in the LL algorithm, since the SoR makes the decision based on the queue length of the server, the SoR has to wait 5s advertisement delay to determine the least load server. Within a 5s time interval, SoR#2 sends packets continuously to only one server, so the queue length of that surrogate server increases gradually. In the meantime, the queue length of the other surrogate server gets decreased gradually. As a result, the LL load balancing method shows a saw tooth pattern for both surrogate servers, as depicted in Fig. 10.

Meanwhile, as depicted in Fig. 11, the RTTs of the RR and RAND algorithms are also low, because SoR#2 redirects the packets to either surrogate server#1 or surrogate server#2 regardless of the advertisement delay since those two methods do not depend on the server queue length. As the server load is properly balanced in

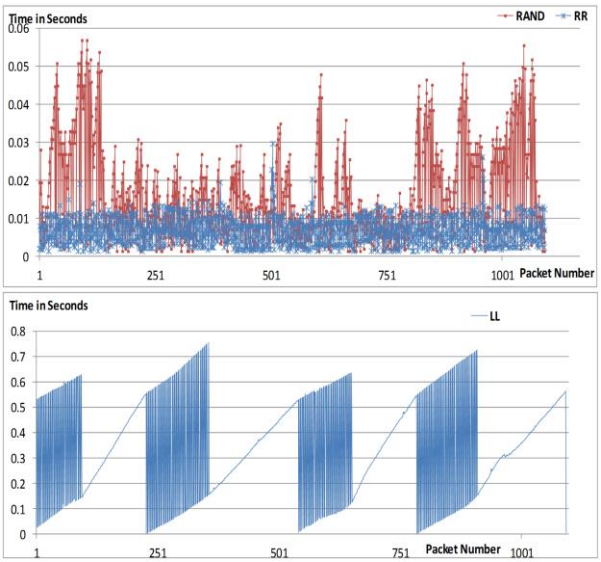


Fig. 11. RTT Measurements for Load Balancing Algorithms

both of RR and RAND methods, server response time is shorter. Therefore, client can get a response packet faster, and the RTT of those packets was low. However, that of the LL algorithm was high. Owing to the longer queue of both servers and the fact that those servers take a long time to reply, as shown in Fig. 11, the RTT is gradually increased within the advertisement intervals. Furthermore, packets have to wait a long time in the server queue and the response time gets increased, so the RTT values were high. In fact, soon after SoR#2 received the server advertisement, it redirected the packets to the other server that has a lower queue value, and the RTT dropped drastically. As a result, the RTT also varies in a saw tooth pattern.

VII. DISCUSSION

Implementation of the SoR as software has both merits and demerits. The main advantage is that it can work as a

router, packet analyser and a packet redirector on different platforms under different Linux-based operating systems. The main disadvantage over this implementation is that hardware acceleration cannot be achieved, because the SoR is a software-based implementation. As Figs. 8-A and 9-A reveal, the SoR consumes more process time than a regular router under some conditions and that affects the RTT eventually. For further optimization, the SoR has to be enhanced with thread-safe functions to obtain better maintenance of the packet queue to reduce the access delays of the queue.

In addition, the proposed system does not maintain a DNS hierarchy, which was needed for managing the locations of surrogate servers in DNS-based RR. This feature figuratively reduces the time consumption to start a connection and to adapt to dynamic changing environments. Additionally, SoRs can capture CDN management messages directly and can also dynamically update their forwarding tables accordingly. As a result, data streams can be rerouted dynamically to an optimized surrogate server without additional costs to clients.

However, performing DPI for both the IP and transport layer headers along with the Layer 7 information increases the SoR workload and the buffer size. As a result, a slight increase in the RTT can be seen in all results compared with the traditional routers. However, we have shown that the RTT delay was almost negligible, around 0.01 s. In terms of QoS and stabilization delay (the most significant point), SoR-based RR showed that it can significantly utilize the delay both in connection initiation and adapt to server changes. Moreover, the table size required for managing content hashes was small, and it supported quick retrieval of next hop IP addresses during packet forwarding.

Furthermore, as contents are frequently changing in industrial CDNs, i.e. game applications, DNS-based RR multi-tires of name servers in order to manage constancy. As a result, in some circumstances, name server communication takes a long time to response. On the other hand, client's TTL values should be set to zero in order to get the frequent updates of the content servers. As a result, client contact local name server frequently to obtain the server changes [1]. But in the proposed method, the SoR communication protocol and the DPI of CDN management packets are able to observe the server changes quickly and update their FIBs according to the changes. As a result, SoRs are at the edges of the network can adapt to the server changes considerably faster and redirect data faster.

Moreover, we presented load-balancing algorithms to test SoR-based RR's suitability as a load balancer.

However, the communication protocol delay is affecting to some load balancing algorithms. Finally, as the Conclusion, the features of SoR, such as passively capturing data, storing data in databases, performing DPI on data from Layer 3 to Layer 7, and making routing decisions based on the packet contents, was utilized to create a prototype for optimization of CDN environments for faster connection initiation and quick adaptation to the server changes.

VIII. CONCLUSION

In this paper, first we implemented UNIX-based software-designed SoR which is capable of analyzing and store necessary data while packet streams are passing through the SoR. Second, we implemented a prototype to demonstrate SoR-based RR to redirect packets based on the content of the packet and the status of content servers. The placement of an SoR as an edge router of the network and programmed redirects packet streams to surrogate servers by analyzing the packet payload showed 50% time reduction in connection initiation and 50% to 60% faster adaptation for the server changes in the network. However all aforementioned advantages came without significant burden to both network and the client, yet the improved flexibility comes at a cost of an extra time of packet analysis on the SoR. That resultant averagely 0.01s increment in RTT of packet transmission.

This paper showed only the quantitative measurement, such as RTT and client waiting time while RR, which is one fundamental component of a CDN network. In future implementations, we will examine the SoR-based CDN infrastructure qualitatively. That includes a discussion about server placement, CDN management and content distribution among servers. Moreover, we will pay substantial attention to proper queue and memory management and implement thread-safe functions to maintain the proper data propagation in software-designed SoRs.

ACKNOWLEDGMENT

This work was partially supported by funds for integrated promotion of social system reform and research and development, MEXT, Japan, by Low Carbon Technology Research and Development Program for "Practical Study on Energy Management to Reduce CO2 emissions from University Campuses" from Ministry of the Environment, Japan and by MEXT/JSPS KAKENHI Grant (B) Number 25280033. Further, this study is supported by Japanese Government (monbukagakusho:MEXT) scholarship program.

REFERENCES

[1] Shaikh, A., Tewari, R., Agrawal, M.: On the effectiveness of dns-based server selection. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1801–1810.

[2] Stamos, K., Pallis, G., Vakali, A., Katsaros, D., Sidiropoulos, A., Manolopoulos, Y.: Cdnsim: A simulation tool for content distribution networks. In: ACM Trans. Model. Comput. Simul, vol. 20, p. 40 (2010).

[3] Nygren, E., Sitaraman, R.K., Sun, J.: The akamai network: a platform for high-performance internet applications. SIGOPS Oper 44(3), 2–19 (2010).

[4] Akamai Technologies: Fast Internet Content Delivery with FreeFlow. http://research.microsoft.com/en-us/um/people/ratul/akamai/freeflow.pdf

[5] CenturyLink, I.: CenturyLink Technology Solutions. http://www.centurylinktechnology.com/

[6] Poese, I., Frank, B., Smaragdakis, G., Uhlig, S., Feldmann, A., Maggs, B.: Enabling content-aware traffic engineering. SIGCOMM Comput. Commun. Rev. 42(5) (2012).

[7] Cohen, E., Kaplan, H.: Proactive caching of dns records: Addressing a performance bottleneck. In: IEEE INFOCOM (2000).
[8] Gadde, S., Chase, J., Rabinovich, M.: Web caching and content distribution: A view from the interior. Computer Communication 24(2), 222–231 (2001).
[9] Koletsou, M., Voelker, G.: The medusa proxy: A tool for exploring user-perceived web performance. In: WCW, Boston, (2001).
[10] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. Maggs, S. Uhlig, V. Aggarwal, F. Schneider, "Collaboration Opportunities for Content Delivery and Network Infrastructures", in H. Haddadi, O. Bonaventure (Eds.), Recent Advances in Networking, (2013), pp. 305-377.
[11] Mao, Z., Cranor, C., Douglis, F., Rabinovich, M., Spatscheck, O., Wang, J.: A precise and efficient evaluation of the proximity between web clients and their local dns servers. In: USENIX Annu. Tech. Conf., Monterrey, pp. 229–242 (2002).
[12] Ager, B., M ̈uhlbauer, W., Smaragdakis, G., Uhlig, S.: Comparing dns resolvers in the wild. In: Internet Measurement Conference, pp. 15–21 (2013).
[13] Kangasharju, J., Ross, K.W., Roberts, J.W.: Performance evaluation of redirection schemes in content distribution networks. Computer Communications 24(2), 207–214 (2001).
[14] Nishi, H: Service-oriented Backbone Router for Future Internet. http: //www.prime-pco.com/4thJEUsymposium/pdf/s1/s1_nishi.pdf.
[15] Inoue, K., Akashi, D., Koibuchi, M., Kawashima, H., Nish, H.: Semantic router using data stream to enrich services. In: International Conference on Future Internet Technologies (CFI08) (2008).
[16] Nishi, H., Kawashima, H., Koibuchi, M.: Information-based Open Innovation Platform. http://openinter.net/.
[17] Mockapetris, P.: RFC 1035, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. http://www.ietf.org/rfc/rfc1035.txt.
[18] Frank, B., Poese, I., an Georgios Smaragdakis, Y.L., an Bruce M. Maggs, A.F., Rake, J., Uhlig, S., Weber, R.: Pushing cdn-isp collaboration to the limit. Computer Communication Review 43(3), 34–44 (2013).
[19] Gummadi, K., Saroiu, S., Gribble, S.: A precise and efficient evaluation of the proximity between web clients and their local dns servers. In: USENIX Annu. Tech. Conf., Monterrey, pp. 229–242 (2002).
[20] Cohen, E., Kaplan, H.: Proactive caching of dns records: Addressing a performance bottleneck. In: IEEE INFOCOM (2000).
[21] Cohen, E., Kaplan, H.: Prefetching the means for document transfer: A new approach for reducing web latency. In: Symposium on Applications and the Internet (SAINT-2001) (2001).
[22] Gummadi, K., Saroiu, S., Gribble, S.: A precise and efficient evaluation of the proximity between web clients and their local dns servers. In: USENIX Annu. Tech. Conf., Monterrey, pp. 229–242 (2002).
[23] Nagatomi, Y., Koibuchi, M., Kawashima, H., Inoue, K., Nishi, H.: A regular expression processor embedded in service-friendly router for future internet. In: Parallel Processing Workshops (ICPPW), 2010 39th International Conference On, pp. 82–88 (2010).
[24] Takagiwa, K., Kubo, R., Ishida, S., Inoue, K., Nishi, H.: Feasibility study of service-oriented architecture for smart grid communications. In: Industrial Electronics (ISIE), 2013 IEEE International Symposium, pp. 1–7 (2013).
[25] Yamaki, H., Nishi, H.: An improved cache machanism for a cache-based network processor. In: The 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (2012).
[26] Hogawa, D., Ishida, S., Nishi, H.: Hardware parallel decoder for of compressed http traffic on service-oriented routers. In: The 2013 International Conference on Engineering of Reconfigurable Systems and Algorithms RSA'13 in WORLDCOMP2013, pp. 3–9 (2013).
[27] Wijekoon, J., Harahap, E., Nishi, H.: Service-oriented router simulation module implementation in fNS2g simulator. The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013), Procedia Computer Science 19, 478–485 (2013).
[28] Sanghi, D.: Unix Socket Programing Computer Networks. http://www.cse.iitk.ac.in/users/dheeraj/cs425/lec17.html/.
[29] Global Inter-cloud Technology Foruml (GICTF). http://www.gictf.jp/indexe.html.
[30] DongJin Lee, N.B. Brian E. Carpenter: Media streaming observations: Trends in udp to tcp ratio. International Journal on Advances in Systems and Measurements No 3 and 4, 147–162 (2010).
[31] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei, "An Empirical Evaluation of TCP Performance in Online Games," In Proceedings of ACM SIGCHI ACE 06, Los Angeles USA, Jun (2006).

Authors' Profiles

Janaka L. Wijekoon received the B.S degree in Information Technology by specializing in Computer Systems and Networking from the Sri Lanka Institute of Information Technology (SLIIT) in 2010. He received his M.S in Engineering Degree from Keio University, Japan in 2013. Currently he is a PhD candidate for the Keio University Japan and his research interests are Future Internet Technologies, Router Architecture Design, Service Based Internet Infrastructure Design, Content Delivery Networks and Network Simulations. Mr. Wijekoon currently is a Japanese Government Scholar (MEXT) for the Hiroaki Nishi Laboratory, Keio University and, he is an assistant lecturer for SLIIT, Sri Lanka and currently on sabbatical leave.

Erwin H. Harahap received the B.S degree in Mathematics from Padjadjaran University, Indonesia, in 1994 and M.S Degree in Computer Science/Integrated Design Engineering from Keio University, Japan, in 2010. Since 2011, he has been a PhD student in Hiroaki Nishi laboratory, Keio University, Japan. His research interests include Network Management System, Artificial Intelligence with Bayesian Networks, Network Modeling, Network Design and Simulations, and Content Distribution Network.

Rajitha L. Tennekoon received his MSc and BSc degrees from Sheffield Hallam University, United Kingdom, in 2010 and 2009 respectively. Since 2013, he has been a PhD student in Hiroaki Nishi laboratory, Keio University, Japan. His research interests include Information Systems Security, Network Design and Simulations, Next Generation Networking and Secured Routing.

Shinichi Ishida received the B.E., M.E, and Ph.D degrees from the Keio University, Japan, in 2005, 2008, and 2013,

respectively. He is currently a project researcher in Nishi Laboratory, Keio University, Japan. His research interests include Future Internet Technologies, Router Architecture Design, Internet architecture and Computer Networks.

Hiroaki Nishi received his B.E., M.E., and Ph.D. degrees from Keio University, Japan, in 1994, 1996, and 1999, respectively. Since 2007, he has been an Associate Professor at Keio University, and since 2010, a Visiting Associate Professor of National Institute of Informatics (NII), Japan. The main theme of his current research is in building of the total network system including development of hardware and software architecture. He places great importance on considering what is required for the highly networked information society in the future. He is investigating the Next generation IP router architecture and Effective Network Systems for Electricity Transaction and Energy Saving in Smart Energy field.

How to cite this paper: Janaka L. Wijekoon, Erwin H. Harahap, Shinichi Ishida, Rajitha L. Tennekoon, Hiroaki Nishi,"Router-based Content-aware Data Redirection for Future CDN Systems", IJCNIS, vol.6, no.7, pp.1-10, 2014. DOI: 10.5815/ijcnis.2014.07.01

