Enhanced Content Navigation Using Edge Routers in Content Delivery Network

A dissertation submitted to
the Graduate School of Science and Technology of Keio University Japan
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Engineering
August 2016

Wijekoon Mudiyanselage Janaka Lankananda

The dissertation is dedicated to my late father Sugath Wijekoon, who is watching over me from the heaven, and to my mother-Jayanthi Jayaweera, who is my lifelong teacher of this life journey. Your motivation, your strength, and your guidance made me who am I today.

Moreover, my Father-in-law Mithila Karunarathna and my Mother-in-law Yamuna Weerathunga, this work is dedicated to both of you. Your understanding of the life taught me living this life effectively.

Nevertheless, the strength of this work is my ever loving wife Nadeesha Wijesundara, this is for you my life partner, my "MANIK."

Chatura and Rajitha, my loving brothers, this is for you guys as well. You both were my arms and you both were my strength in Sri Lanka during the period of this study.

Thank you all for keep me smiling, never ending inspirations and, above all, for your always inspired motivation and tired patience.

Finally, this work is dedicated to all my teachers who always gave me advices to become a successful man.

We Did It!!!

Abstract

The Internet can be defined as a network composed of geographically dispersed servers and clients. In principle, clients request content from servers, and the servers respond to the requests by sending the requested content to the clients. The content should be navigated among networks, and certain rules and methods have been developed to achieve optimized navigation. Navigation is definable as the process of finding a destination and reaching that destination using a preferable route. Hence, the main challenges for achieving content navigation on the Internet can be summarized in the following two directions: 1) to determine and select service points and 2) to route users to selected service points.

The need for optimized content delivery accelerates the development of the Internet by proposing content delivery networks (CDNs). CDNs use content cache servers within Internet Service Provider (ISP) networks and select a service point for a content request of a client by using the Domain Name Service (DNS). ISPs, on the other hand, can route a client to a service point according to a selected network path by using routing protocols, which are optimized based on the link state information. Namely, content providers and ISPs are separated in content navigation. Thus, researchers are proposing that "the effectiveness of content navigation is doubtful in the absence of a reliable collaboration between the ISPs and CDNs." Meanwhile, network device manufacturers have been upgrading servers, routers, and links to provide innovative services to enhance content navigation.

To this end, dissertation proposes an approach to enhance content navigation in CDNs by using edge routers of ISPs. Edge routers can be utilized to create a collaboration between ISPs and CDNs by collecting and using both the network state information of the ISPs and the content server state information of the content providers simultaneously to leverage content navigation in a CDN. Dissertation proposes a solution for the collaboration by using a Service-oriented Router (SoR) as an edge router. The SoR is a novel router architecture for providing content-based services by shifting the current Internet infrastructure to an information-based open innovation platform. SoR uses the Server Link Router-state Routing (SLRouting) protocol to collect both network and server state information. The SLRouting protocol hypothesis is a new paradigm of network path selection using network device states to calculate the network path selection metric. SoR utilizes the collected information for selecting a service point that is appropriate to a user request and routes the request to the selected service point by leveraging the DNS-based user redirection and by performing content-aware packet redirection.

Consequently, the structure of the dissertation is divided into three main sections: 1) design and development of a software SoR, 2) design and implementation of the SLRouting protocol, and 3) the use of both SoR and the SLRouting protocol to enhance content navigation by creating a collaboration between ISPs and CDNs. In addition to the evaluation of the software SoR and SLRouting protocol as individual units, dissertation implemented an emulator-based environment using Planet Lab and a

simulator-based evaluation environment using ns-3 for evaluating the proposed CDN architecture supposing a wide area network. In conclusion, the proposed architecture yields better performance in terms of request redirection and effective network resource utilization, and serves as a guideline for future content service models by addressing adequate ISP-CDN collaboration through enhanced content navigation.

Acknowledgements

At various stages of proposal, design, implementation, evaluation, and writing the dissertation, I had interesting discussions, support, and valuable feedback from my supervisor Prof. Hiroaki Nishi. I would like to express my sincere gratitude to him for the valuable guidance and the support throughout these years.

I would like to give my heartfelt gratitude to Mrs. Yuko Nishi for the endless support during this study. Moreover, I would like to acknowledge all West Laboratory members, especially Erwin Harahap, Rajitha Tennekoon, Kenichi Takagiwa, Shinichi Ishida, Fumito Yamaguchi, Tomoya Imanishi, Sota Sawaguch, and Tianmeng Shen for great support and help during the time I spent in West Laboratory. I would like to acknowledge my friends Kasun Prasanga, Kazuki Tanida, Prabhath Buddhika, Windhya Rankothge, Udara Rajapaksha, Tharindu Nanayakkara, and Kalpani Manathunga for the great support, you all were "always there" for me. A very special "thank you" goes out to "my strength," my family, for their understanding and support throughout the years, which, by the nature of happenings, meant sacrificing many years and waiting to see my success; nonetheless, they never stopped cheering me on.

I also want to acknowledge the following people and institutions for the precious contribution to the success this thesis work.

- Dear Dr. Pradeep Abeygunawardhana, for giving me this opportunity to study in this prestigious university and always valuable advises and the motivations to become the person who I am today.
- MEXT (Ministry of Education, Culture, Sports, Science, and Technology), for full financial support throughout the period at Grad. School of Science and Technology, Keio Univ., Japan.
- Prof. Iwao Sasase, Prof. Fumio Teraoka, Prof. Hiroshi Shigeno, and Prof. Michihiro Koibuchi for the valuable comments, guidelines, and, above all, for the advices to make the dissertation a success.
- Planet Lab, RocketFuel, and WIDE network for research support in network typologies.
- ns-3 mailing list for the support during the ns-3 implementations.
- KLL research grant and Keio University Doctorate Student Research Grant for the support during the thesis study.

Last but not least, I would like to give my sincere gratitude to SLIIT, dear Dr. Malitha Wijesundara, and dear Prof. Koliya Pulasinghe to keep me motivated during my higher studies.

Janaka L. WIJEKOON

August, 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The motivation

In the 1930s, mathematician and pioneering computer scientist Alan Turing described the principle of the modern computer in [1]. A few decades later, with the acknowledgment of Von Neumann's central concept of the modern computer and the evolution from vacuum tubes to Integrated Circuits (ICs), digital computers started to evolve in the 1950s. Since then, numerous institutes, researchers, and scientists have been interested in developing computer-based technologies. Among them, some have been interested in developing technologies to communicate among computers [2]. In this spirit, Licklider, Ivan Sutherland, Bob Taylor, and Lawrence G. Roberts convinced the Defense Advanced Research Projects Agency (DARPA) about the importance of computer networking concepts. In late 1966, Roberts initiated the ARPANET project and published ARPANET, which is considered the origin of today's Internet [3].

In the midst of this evolution in the 1990s, the Wide Area Information Server (WAIS) [4] project, Archie search engine [5], and Gopher protocol [6] were developed and coexisted for some period as the Internet. Later, those systems were incorporated into yet another system, the World Wide Web (WWW) [3]. WWW provides an architecture to connect and access data, i.e., content from networked computers, the Internet. As illustrated in Fig.1.1, WWW assumes that the content is placed in distant locations, i.e., servers and clients should contact servers to access the intended content [7]. That process was developed based on two fundamental methods. The first one is to find the location of the server, and the second one is to reach the server and access the content. Content typically is represented in the form of Web pages, which contain texts, graphics, multimedia, and hyperlinks. Meanwhile Transmission Control Protocol (TCP)/Internet Protocol (IP) protocol stack was gaining momentum since the early 1980s [8]. Consequently, IETF has proposed protocols such as the Hyper Text Markup Language (HTML) [9], Hyper Text Transfer Protocol (HTTP) [10], Universal Resource Locators (URLs) [11],

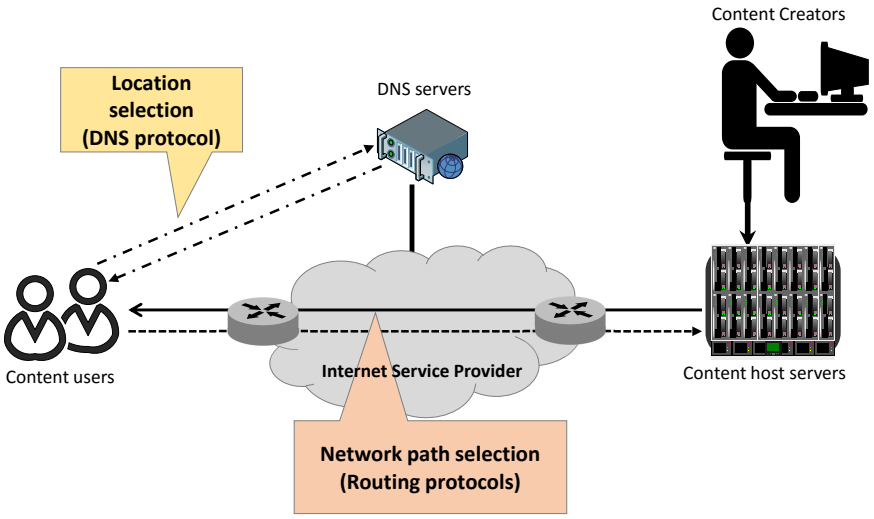


FIGURE 1.1: Client-server communication via an ISP

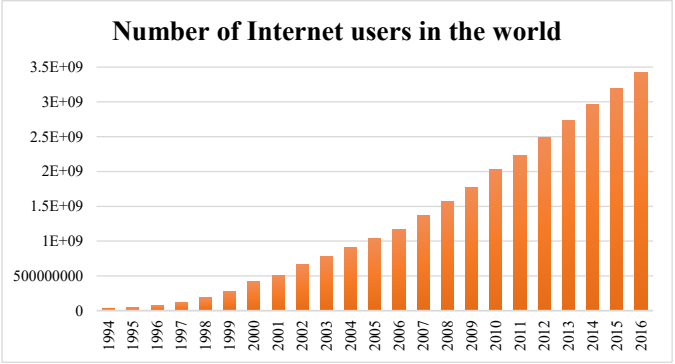


FIGURE 1.2: Growth of the Internet users within last two decades [12]

Universal Resource Identifiers (URIs) to standardize clientserver communications. Such protocols improved the Internet as the principal method of communication since the late 1990s.

The concept that "information sharing via networks is becoming the driving force of the future Internet" has boosted the growth of the Internet traffic with the evolution of WWW. Moreover, the rapid acceptance of broadband services, along with content richness and the need for complex systems has resulted in an increase in the number of Internet users as depicted in Fig.1.2. This very nature of traffic growth brought up challenges in content delivery. In fact, the degraded quality of service (QoS) such as increased delay in content delivery led people to redefine "WWW" as "World Wide Wait." However, researchers and scientists continuously researched new technologies such as network architectures, applications, and software tools to improve content delivery and services over the Web. Alternatively, as presented in [13], a combination of these technologies was used called "content networks (CNs)." A CN can be defined as a communication network that deploys an infrastructure of components at protocol layers four through seven in the ISO/OSI reference model [8, 14]. These components are interconnected and interact with each other, creating a virtual network layer on top of an existent packet network infrastructure, i.e., over a service provider network [8]. Several terms and names have been

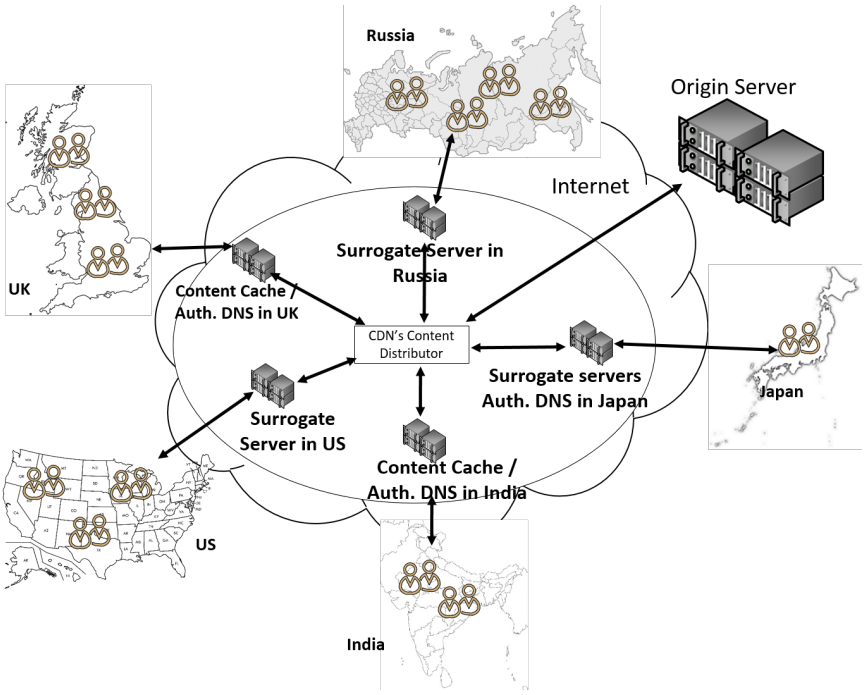


FIGURE 1.3: A traditional Content Delivery Network

proposed and used to popularize CNs, among which "Content Delivery Networks (CDNs)" are widely used among researchers and business bodies [14]. Figure 1.3 illustrates a conventional CDN. As illustrated in the figure, a CDN is a globally distributed network of content servers i.e., reverse proxy servers deployed in multiple data centers over several geographical regions.

In general, a CDN consists of two major business bodies:

1. Content Providers (CPs)
2. Internet Service Providers (ISPs)

Characteristically, CDNs are introduced to ensure high availability and performance by placing content servers near the users. Very recently, ISPs have begun inviting CPs to deploy their networks within the ISP networks [15]. The reason is that ISPs own the last mile of the Internet, and they cache the content deep in their network so that the distance to be travelled by the requests can be further reduced. Furthermore, there is an emerging trend among Telecommunication Service Providers (TSPs) to host their own CDNs, telcos [16]. Consequently, one of the major CDN providers, Akamai [17], claims that they are hosting more than 80,000 servers within over 1,100 networks in more than 1,800 locations spanning 80 countries. Furthermore, they claim that they are delivering more than 20% of the Internet traffic [18].

In principle, as illustrated in Fig.1.1, regardless of the network infrastructure, i.e., WWW or CDN communication between users and servers is defined by two significant steps:

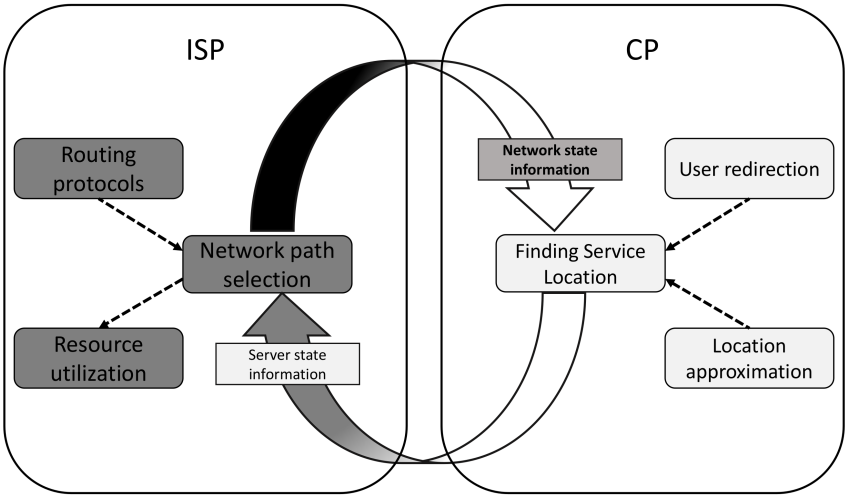


FIGURE 1.4: Inter dependency of ISP and CP for content navigation

1. Determine and select the location of service, i.e., servers.

2. Route user requests to the location of service, i.e., network path selection.

The above two steps can be summarized as the process of "Content navigation" across the Internet [14]. In detail, content navigation can be defined as the problem of determining the location of the service, routing packets to the adequate destinations across the Internet, switching packets to the most effective server, and finally returning the response to the user.

In general, the CPs, which host Web servers and gather content from content generators such as Facebook [19], solve the server selection problem [20]. CPs use Request Redirection (RR) techniques to offer the best servers to their subscribers by considering one or possibly more criteria such as distance and server status [14, 20]. Among many protocols, CPs find the best server for the users by mainly using the Domain Name System (DNS) [21] according to the criteria mentioned above, which is also known as DNS-based user redirection [14]. ISPs, which are the connection providers of the users, solve the routing problem. ISPs use IP Layer (L3) network path selection methods to maintain the communication between users and servers. Commonly, ISPs use routing protocols such as OSPF [22], ISIS [23], EIGRP [24] and RIP [25] to solve the network path selection problem.

Solutions to the abovementioned two problems are the primary challenges of effective content navigation. However, when CDNs are deployed within the ISP networks, both CPs and ISPs face several additional problems. The first problem is "are generic routing protocols capable of selecting network paths according to the dynamicity created by the CPs?" The second problem is "how do ISPs consider the dynamic behavior of the CPs for network path selection?" The third problem is "how do CPs use ISP network state information for service point selection?" Thus, as illustrated in Fig.1.4, server selection and network path selection are no longer two different problems solved by CPs and ISPs individually [26]. Both steps are interconnected and influence each other [26]. Therefore, the

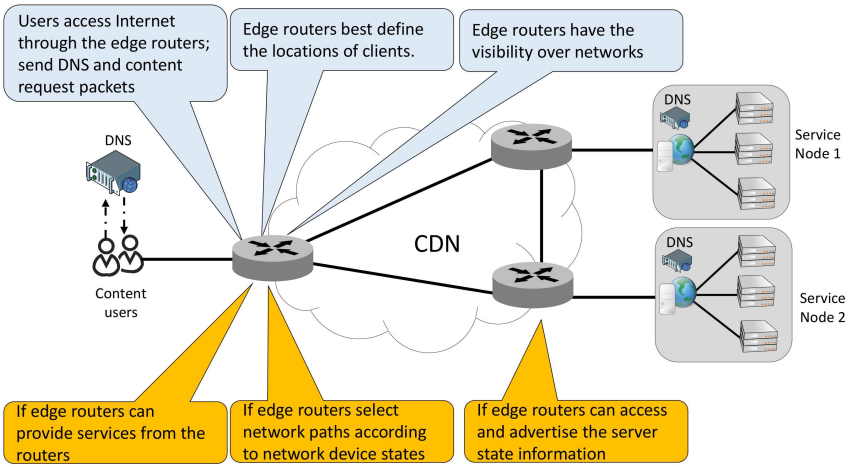


FIGURE 1.5: Edge routers of a CDN

effectiveness of appropriate server selection and network path selection is questionable in the absence of reliable cooperation between server selection and packet routing, i.e., between ISPs and CDNs. For example, although Google claims that they maintain their CNs over several ISPs, [12] states that on average, a Google query must travel approximately 1,500 miles to a data center and return to the user. In essence, effective content navigation can be achieved when ISPs and CPs share their information for server selection and network path selection [27].

According to the basics of computer networks, internetworking defines the process of linking computing devices and users through a maze of communication lines. As explained in [28] and depicted in Fig.1.1, network routers play a major role in computer networks. Routers, as a gateway to both the Internet and CDNs, interconnect networks and forward information/packets towards their intended destination [28]. Network routers are gaining momentum because they can be used not only for the conventional "packet forwarding" function but also to provide application services [29]. Cisco, Juniper, and Qualcomm are introducing new router architectures and proposing placement of their routers at the edges of networks to provide user services such as Deep Packet Inspection (DPI) [30], load balancing [31], DDoS prevention, Session Border control [32], and WAN acceleration [33]. Hence, edge routers on both the users' edge and CPs' edge are becoming imperative devices that can work as intermediaries to solve the content navigation problem including both network path selection and server selection.

As depicted in Fig.1.5, edge routers of a CDN i.e., gateway routers are vital because all users and local DNS servers access the Internet via the edge routers. Nonetheless, the edge routers define the location of the users because the gateway routers connect the users and the Internet. Moreover, the routers have visibility over the network, and the routers know the up-to-date topology information of the ISP. As previously mentioned, edge routers are the primary consideration of router manufacturers to provide effective user services [29–33]. Consequently, if the edge routers can provide user services, they can select the network paths according to the network device states and can access the server state

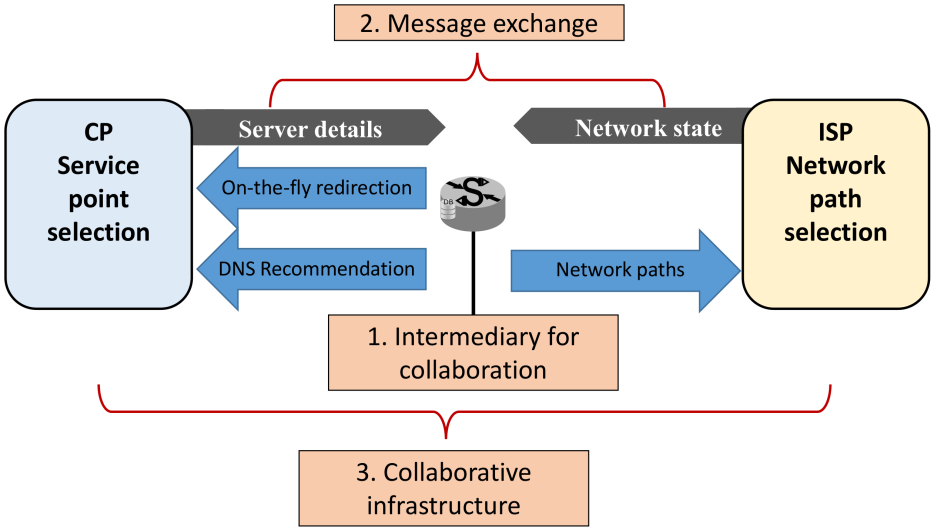


FIGURE 1.6: Main three objectives of the dissertation

information and advertise it among the other routers in the network; the edge routers of a CDN can be used as intermediaries to presume a collaboration between the ISPs and the CDNs.

To this end, as illustrated in Fig.1.6, the main objective of this dissertation is to use edge routers to solve the utterly significant problem of content navigation within CDNs. Thus, the dissertation proposes to utilize edge routers of a CDN as intermediaries to build a collaboration between an ISP and a CDN. Edge routers maintain a protocol to collect and use network-state information and CDN server state information to solve the server selection problem by leveraging the DNS-based user redirection; IGPs are used as the message exchange protocol. In essence, as presented in Fig.1.6, the dissertation contemplates three proposals to exploit edge routers of the CDNs to solve the content navigation problem by creating a collaboration between network path selection and server selection.

1. **Select the intermediary for collaboration:** given that the regular routers are still incapable of providing user services, dissertation proposes to use the software Service-oriented Router (SoR), which is a novel router architecture introduced to provide services to end users [34, 35], as the edge routers of CDNs.

2. **Use IGP as the message exchange:** the dissertation proposes a novel routing paradigm, which adequately selects the network paths according to the states of network devices such as routers, links, and servers to solve the routing problem, called the SLRouting protocol.

3. **Implement the collaboration infrastructure:** use both SoR and the SLRouting protocol to enhance the content navigation in an infrastructure by creating a collaboration between an ISP and CDN. Finally, the dissertation implements a prototype of the proposed collaborative approach.

Figure 1.7 illustrates the places addressed by the dissertation in the domain CDN and ISP. As depicted in red, the dissertation primarily proposes an infrastructure that maintains a collaboration between ISPs

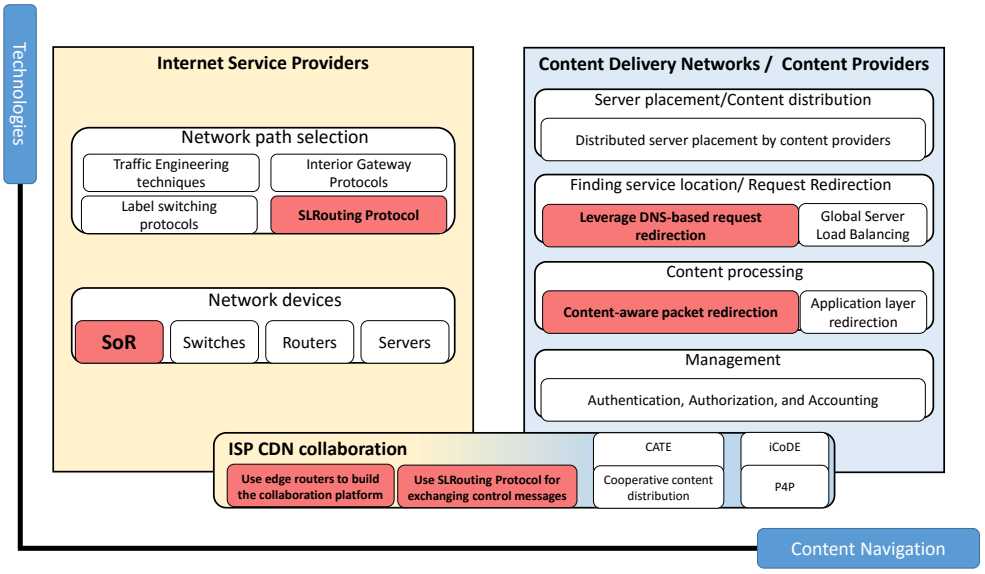


FIGURE 1.7: The place the dissertation resides within the current context

and CDNs to enhance content navigation. The SoR is then proposed as the intermediary between the ISP and CDN to facilitate the collaboration. Moreover, the SoRs are used to leverage the commonly used CDN server selection technology, DNS-based redirection, according to both network and server state information. The proposed CDN infrastructure presumes that the SLRouting protocol selects the network paths according to the network device state information and content server state information. Moreover, the SLRouting protocol is used by the proposed CDN architecture to advertise the server state information and distribute it among the rest of the SoRs in the CDN.

1.2 Dissertation structure

Figure 1.8 elaborates the structure of the dissertation, and Table 1.1 briefly describes each chapter; each chapter has a purpose, objective, proposal, and achievement. As displayed in Fig.1.8, Chapter 2 explains the background study related to the dissertation and provides detailed explanations about the evolution of CDNs, RR methods of the CDNs, and the limitations of those approaches. Moreover, Chapter 2 explains some related work about router architectures designed to provide application services. Furthermore, improvements required for existing routing protocols to solve the content navigation problem are described. Chapter 2 also provides some related research works that pointed out the necessity of ISP-CDN collaboration, that address the content navigation problem in several different perspectives. Chapter 3 proposes a software SoR because the SoR continues to be a topic of research and development. Moreover, its software and hardware components are currently being developed to create a complete router. Chapter 3 explains the software SoR, which is designed using network simulator-2 and -3 (ns-2 and ns-3). Chapter 4 proposes a novel routing paradigm, Server Link Router-state Routing Protocol (SLRouting), that selects the minimal delay network paths according to the

network device state information. The proposed protocol is evaluated using Exodus-USA (AS3967) and Telstra-AUS (AS1221). Chapter 5 explains how both SoR and the SLRouting protocol are used in a CDN infrastructure to enhance content navigation by forming a collaboration between ISPs and CDNs, particularly regarding how the edge SoRs are used for information collection, information sharing, location detection, and solving the content navigation problem. Moreover, this chapter describes message flow among users, edge routers, and CDNs for the scenarios of DNS recommendation and content-based user redirection. The collaboration architecture is evaluated using two prototype implementations. The first prototype is implemented using the PlanetLab environment, and the second prototype is implemented on the ns-3 simulator using the WIDE network's topology. Finally, Chapter 6 concludes the dissertation by noting future visions.

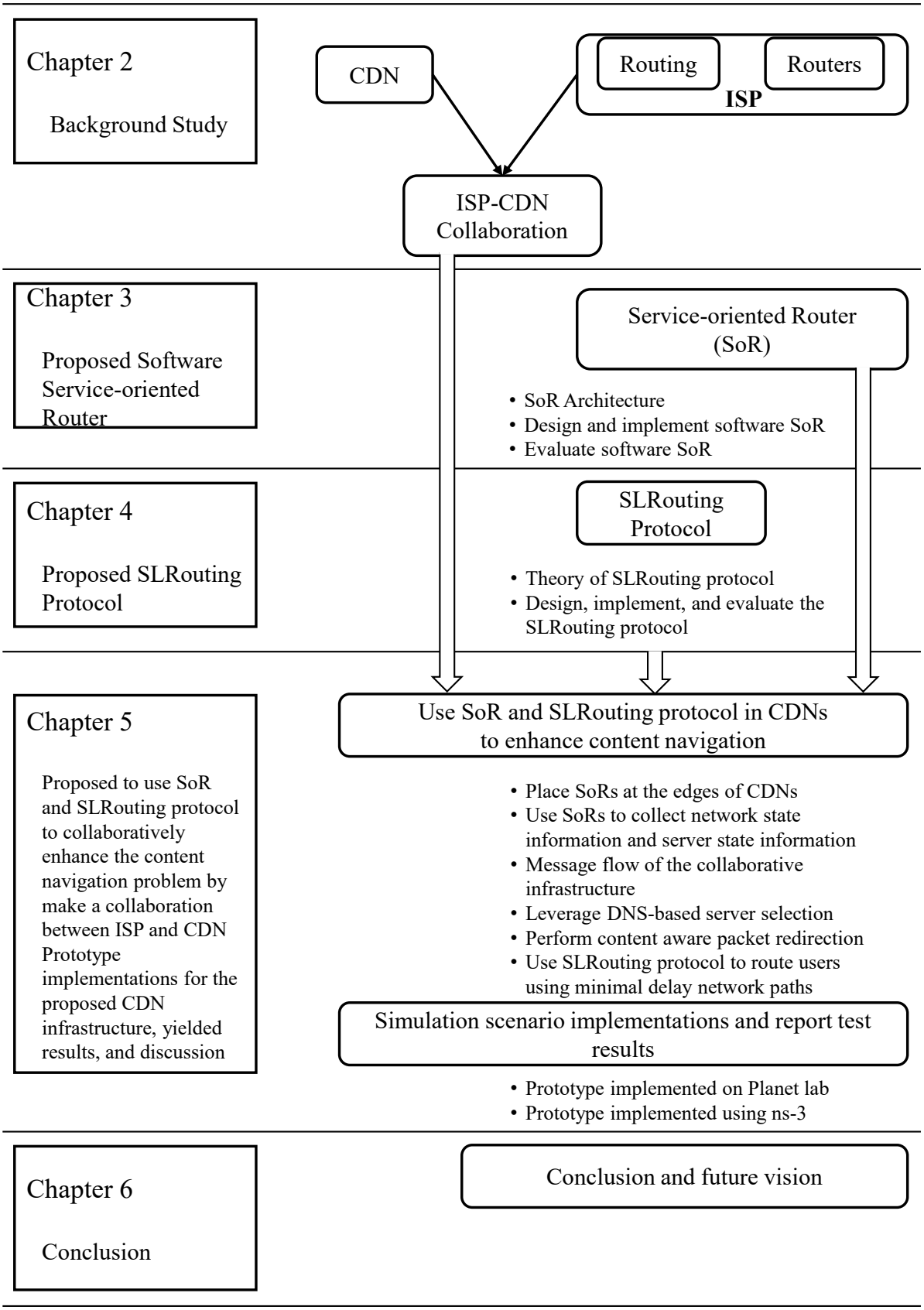


FIGURE 1.8: Dissertation structure

TABLE 1.1: Chapter description

Chapter 2	Purpose	Survey the background information about IPS, CDN, content navigation, existing technologies for content navigation, the necessity of ISP and CDN collaboration to enhance content navigation, and existing attempts regarding ISP and CDN collaboration.
Chapter 3	Purpose	Proposed Software Service-oriented Router (SoR)
	Objective	1. Find a simulator to design and implement the software SoR 2. Design and implement the software SoR similar to the original SoR
	Proposed Method	Use ns-3 and ns-2 simulators to implement the proposed software SoR. Design modules to perform routing, packet analysis, and recommendation, and store packet information in the software SoR.
	Achievement	Designed and implemented SoR in both ns-2 and ns-3 simulators. Tested both software SoRs for functionality and effectiveness. The implementations are openly published in [36–38].
Chapter 4	Purpose	Propose a routing protocol to select network paths based on the states of the network device i.e., servers, links, and routers.
	Objective	1. Represent network devices on a common scale to build the route metric i.e., cost 2. Consider the burden made by content providers to select the network paths 3. Use necessary techniques for stable route propagation and route management
	Proposed Method	Represent network device states on a common scale, "time," and thereby calculate the route metric, i.e., cost. The proposed protocol calculates the cost based on the end-to-end propagation delay and selects the minimal delay path between two destinations.
	Achievement	Designed and implemented the Server Link Router-state Routing (SLRouting) protocol, which is a hybrid IGP that selects network paths according to the network device states. The protocol displayed more than 90% network resource utilization. Moreover, 2030% propagation delay reduction compared to OSPF was achieved.
Chapter 5	Purpose	Propose using both SoR and SLRouting to enhance content navigation by creating a collaboration between ISPs and CDNs.
	Objective	Use both SoR and the SLRouting protocol to use network device state information and server state information to leverage DNS-based user redirection. Moreover, use the network device state information for adequate network path selection to encase the content navigation.
	Proposed Method	1. Place SoRs at the edges of the ISP i.e., gateway routers and use the SoRs to collect network device information and server state information 2. Use the SLRouting protocol to advertise both network state information and server state information among the SoRs in the CDN 3. Use SoRs to select less busy servers, and forward the user request to the selected servers using minimal delay network paths by solving the major two parts of the content navigation problem
	Achievement	Successfully implemented the ISP CDN collaborative platform to encase content navigation by leveraging DNS-based redirection and selecting minimal delay network paths. The proposed method reduced the data download latency by an average of 40% and jitter by an average of 40% compared to conventional DNS-based redirection. The collaborative approach reduced the user redirection time into small time scales of milliseconds. Moreover, the proposed method utilized all links in the network by marking 100% network resource utilization.

Chapter 2

Background study and related works

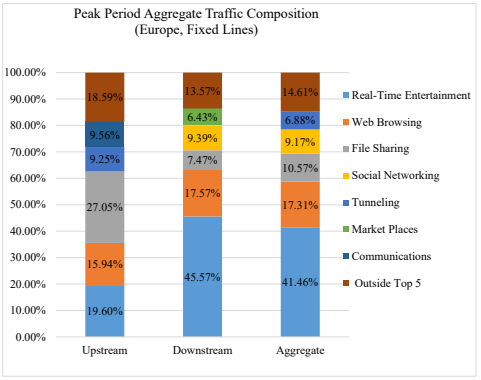
2.1 Content navigation in WWW

Transport requires navigation. Navigation is the key principle of finding a destination and travel towards it. Travel is erratic and mostly uncertain until a path towards the destination is found and known. Given the Internet architecture is a giant web of clientserver communication and the servers are placed in remote locations, clients should navigate their data across the Internet to reach the intended server. Similarly, servers should navigate the requested content across the web to make sure the content have been reached to the intended user. The responsibility of content navigation lies upon the Internet Service Providers (ISP) according to the traditional client-server communications. As pointed out in Chapter 1, the content navigation problem can be divided into major two parts:

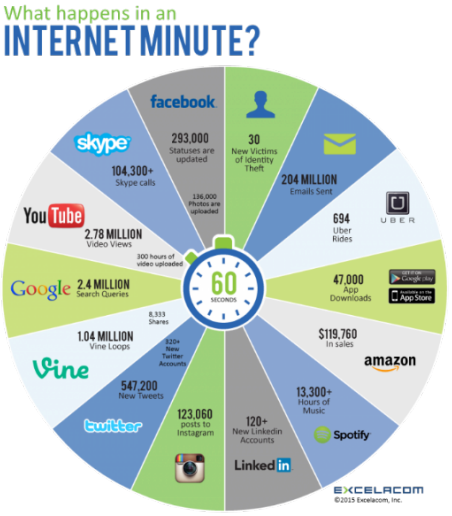
1. Select service points
2. Select network paths to reach the selected service point

Technically the service point selection problem can be divided into two parts. The first part is selecting a location of a service point; the service point could be a data center where users can find the content servers. The second part is to find a server and download the data. Generally, in traditional WWW, content providers solve the service point selection problem; web hosts use Domain Name System (DNS) [21] to let clients know the service points. When users reach their service points, the data centers use several technologies and select the intended server to fetch the data. Consequently, data centers use the reverse proxy technique [14], caching techniques [14], software defined network concepts [39], and load balancing [40] techniques for selecting and forwarding users to the content servers.

The network path selection is the ultimate responsibility of ISP networks [8, 41]. When a user requests either DNS message or content request message from the ISP through their gateway router, ISPs determine the network path, which can be used to reach the intended destination, and forward the packets



(I) Peak Period Aggregate Traffic Composition - Europe, Fixed Access [42]



(II) Internet within a minute [43]

FIGURE 2.1: Internet, now and towards its future

towards the intended destinations. The feasible paths can be itemized as shortest path, and the shortest paths are calculated according to several criteria such as paths with a minimum number of hops, paths with the best link details, or the paths with minimal delays [8, 41]. Consequently, ISPs use routing protocols to find the best path to route the users to their intended destination [8, 41]. Furthermore, when the servers create the reply packets with the intended content for the users' requests, again, it is the responsibility of the ISPs to route the content towards the correct users. Overall, in the traditional web services, the content navigation problem was solved by web hosts and the ISP with minimal interaction with each other.

Since the late 1990s, the WWW evolved from an Internet application for scientists and researchers to the world's largest communication platform as today. Almost all businesses and companies started to evolve with the WWW and use the feasible features of the Internet to develop their businesses. Companies and business depend more than ever on the Internet to instantaneously deliver relevant content and services to their users. However, the massive growth of the Internet makes the web system distributed and decentralized [8]. Consequently, scalability and the delay issues were raised on the Internet, and the WWW term was interpreted as "world wide wait" system. Web hosting companies added more bandwidth, hardware upgrades, and they upheld lots of techniques to maintain quality-of-service (QoS) [8] as one potential remedy of increasing the performance of the websites. However, such approaches did not sustain a long term because the growth rate of the Internet became exponential due to the popularity of Internet access technologies such as broadband connections [42, 44].

In essence, effective content navigation incurred both content providers, i.e., web hosting companies, and the ISPs lots of challenges. Various methods were proposed to solve the content navigation problem by ISPs and the web hosting companies. Stablishing server farms, caching content in intermediary nodes [45], place same content over multiple web servers, and load balancing the servers [14, 15, 45]

TABLE 2.1: Evolution of CDN [48]

	1st Gen. CDN (static CDN)	2nd Gen. CDN (Dynamic CDN)	3rd Gen. CDN (Multipurpose CDN)
Content served	Static HTML and downloadable files	Static and dynamic content, including rich media	Static and dynamic content, including mobile and rich media
Caching method	Origin push	Many are origin pull	Most are origin pull
Network topology	Scattered	Consolidated	Highly consolidated
Agenda	Performance	Performance and availability	Security, performance and availability
Pricing	Very expensive	Expensive	Affordable
Customer base	Corporate sector	Business sectors	Anyone with a website

are some attractive solutions proposed by the web hosting companies. Deployment and the growth of the server farms normally go hand-to-hand with the appropriate upgrades of the network links that connects the websites to the Internet. As a consequence, ISPs used techniques such as increase network bandwidth, introduce backup path selection technologies [46], improve the capability and the processing power of the routers [29], and use optical communication media to cater the requirement of the content navigation.

As presented in Fig.2.1, the percentage of dynamic content on the web is increased, and the users demand more services from the web server. Therefore, it is no longer sufficient to distribute just static content. Instead, recent developers are extending the idea of a distributed content model to include the services operating on such content as well, i.e., dynamic delivery network (DDN) [47]. Consequently, numerous architectures and systems are being developed to move server-side services to the edge of the Internet closer to the users. This concept is eventually evolved as the "Content networks."

2.2 Evolution of the content delivery networks

A scalable approach to overcome the bottlenecks of the web systems and slowing down the content navigation is to move content near to the users where it becomes faster to retrieve [14, 15]. Users' requests are then redirected and served from the devices which are placed near the users. Server replication [14, 45] and proxy caching [14, 45] are two technologies used to move content near the users. Namely, these two technologies are the main building blocks for the content delivery networks (CDNs). The proxy caching can be interpreted as a device that stores the requested web objects in an intermediate location, which is preferably close to the client. A web cache resides between the web servers. i.e., origin servers and one or more web clients, and the content requests are monitored as they come by. Web cache saves a copy of those content for itself, and if there is another request for the same object, the web cache uses the copy that it has instead of forwarding the request to the origin

server. Caching web objects is an interesting topic of research and development. Thus, Rabinovich et al. provide a detailed information about web caching [45].

Table 1.1 summarizes the evolution of CDN. Since this is the third generation of CDNs, let's consider the CDN topologies used in third generation CDNs (3G CDN). As pointed out in [48], there are two different content topologies available in 3G CDNs. The first type is scattered CDNs; scattered CDNs operate a high number of medium and low-capacity point-of-presences (PoPs), which densely populate among selected geographic regions. Such topologies are focusing on optimal physical proximity. Consequently, in this model, sometimes, PoPs are positioned very close to one another. Early 3G CDNs, deployed during a transition period between copper and fiber wiring, relied on the scattered model. With time, as more fiber cables were laid down, and thus, global connectivity continued to be improved, the marginal benefit of minimizing the physical distance to servers continued to diminish. Moreover, as CDNs continued to introduce more customization features, the scattered topology displayed delays in systems responsiveness, thereby preventing rapid configuration deployments [48].

The second one is consolidated CDNs. Consolidated CDN operates a small number of high-capacity PoPs, which are strategically positioned in major data centers, to serve a wider population. This network topology represents a more modern approach to content delivery that was made possible by the evolution of Internet connectivity [48]. The main benefit of a consolidated topology is its centralized infrastructure, which enables agile management and rapid configuration deployments. This benefits both end users and the network operator, offering more control and better overall responsiveness. One example for consolidated CDNs is the Telcos [16]; Telcos are the CDNs which are deployed within the local ISP networks and cache content deep in the ISP networks.

However, with the evolution of such content delivery networks (CDNs), the content navigation problem became more complicated [15, 49, 50]. Nonetheless, several new questions were introduced to the content navigation problem such as:

1. Where to place the content?
2. How to select the nearest service point to the users?
3. How to notify the users about the selected service points?
4. How to select network paths according to the states of the network devices?

Above all, the question, "are existing technologies such as DNS and shortest path routing protocols capable of solving the continent navigation problem in CDNs," became the utterly important consideration of researchers, developers, and business bodies. Following subsections provides basic descriptions of the technologies potentially answer the first three questions above. The answer to the last question is described in-detail in Chapter 4 Section 5.

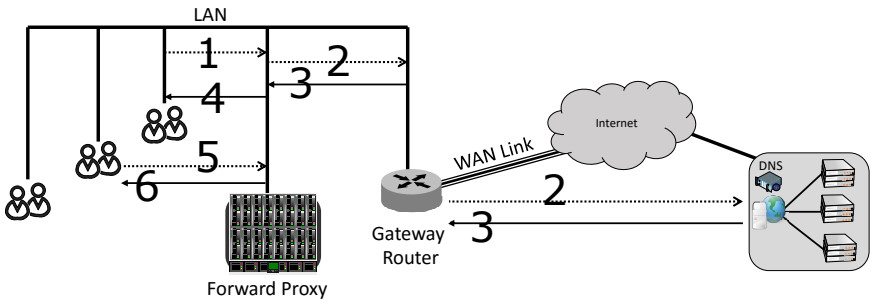


FIGURE 2.2: Forward proxy concept

2.2.1 Place content caches in networks

"Where to place the content" is determined by the popularity of the PoPs and the density of the users [14, 15]. In general, the content placing is identifying as "placing a content cache in the network." There are three shared content caches available in the network.

1. A Forward proxy: a proxy server works on behalf of a specific group of content consumers.
2. A Reverse proxy: a proxy server works on behalf of the origin server and helps to deliver content using group of servers. The reverse proxy servers are also known as a server accelerator.
3. Interception proxy: An interception proxy server directs the traffic to a web caches which might place near the users.

Figure 2.2 displays a scenario of a forward proxy. As illustrated in the figure, forward proxy servers are placed within the local networks. Most of the applications support forward proxy [51]. Forward proxy provides the workgroup network with several advantages. It is likely that a group of users, i.e., universities, have common interests and visits the same websites several times because they work together. This increases the probability of accessing the same content over and over again. Therefore, if the content is cached within the network when the content is fetched at the first place, next time the content can be fetched from the local network itself. This scenario drastically reduces the content navigation latency. This concept works well in the work group networks because the content fetching is easy. One of the main limitations of this method is that the applications have to be manually configured to identify the proxy. However, there is a method that can be used to auto-configure the proxy setting, i.e., dynamic host configuration protocol (DHCP) [52]. As a matter of fact, most of the modern CDNs also uses the same technique to cache the dynamic content within the ISP network [14]. As presented in Table 2.1, the content cache uses a pull method from the main servers when the content is required [48]. However, the main limitation is "method of notifying users about the content cache placement." For that, there is no automatic method.

A reverse proxy is the main building block of content delivery networks [14]. The reverse proxies are also commonly known as server accelerator, and sometimes, gateways of a server farm. Figure

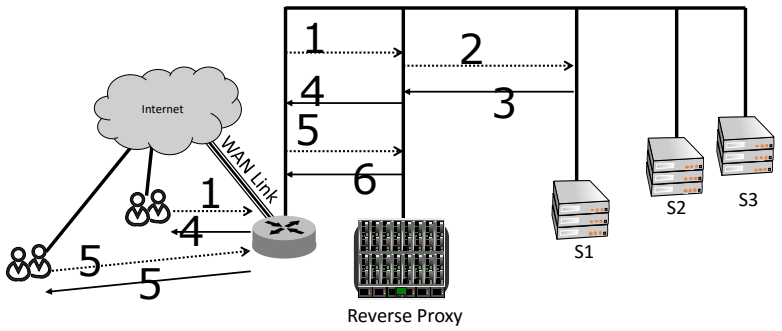


FIGURE 2.3: Reverse proxy concept

2.3 displays a simple scenario of a reverse proxy. Reverse proxy gives the content provider several advantages. The proxy gently reduces the traffic load on the origin servers. Since the full extent of the content of all the site's origin servers is limited, the probability of a cache hit is quite large [14]. This enables the servers to be optimized for creating and editing content, rather than for serving heavy volumes of the traffic to the network. Thus, it also improves the scalability of the site. If the content demand is increased, more servers can be added to the system as accelerators. Moreover, the origin server can be placed in the distant locations, and the content can be pulled from the origin server when required. That is the main reason this concept eventually evolved as the basic concept of the CDN. However, as pointed previously, even though this method can place content near their customers, the main limitation is letting users know the location of the server accelerator.

As a consequent, users should find the server accelerator first, after that, server accelerator finds an adequate content server for the users. In technical terms, determining the location of service in a CDN is referred to as "switching," and network path selection and navigation through the network are referred to as "routing." In traditional networks, switching is the term used to distinguish several local endpoints connected to a Layer-2 switch [8, 14, 41]. However, when coming to CDNs, the switching becomes sophisticated, and the switching refers to service point selection based on Layer 4-7 (IP layer through application layer). Routing remains same as selecting network paths based on Layer-3 network path selection.

2.2.2 Determine the service location

There are several techniques used by the CDNs to determine the service location and direct client request to the selected service location: Global server load balancing [14], domain name system (DNS)-based request redirection [53], HTML rewriting [14], and anycasting [14]. DNS-based request redirection is the most commonly used method by the CDNs to redirect the client requests to the service point due to the advantages the DNS protocol provides, e.g., protocol independent. Moreover, other than the anycast method, all other methods are heavily contingent of the DNS-based method and those

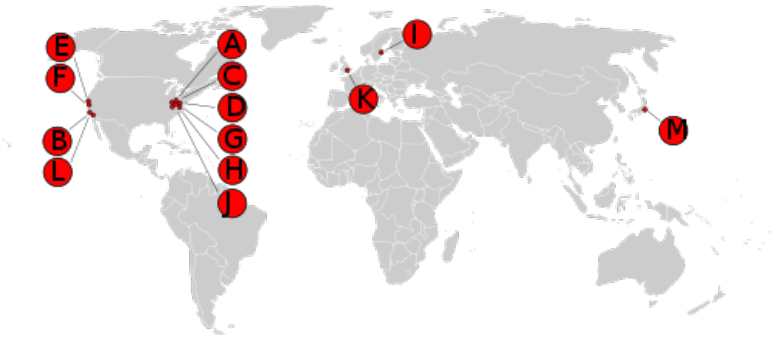


FIGURE 2.4: Map of DNS root servers
(M is WIDE project, M.ROOT-SERVERS.NET)

methods are usually deprecated by the DNS-based request redirection. Therefore, dissertation proposes a method to leverage the DNS-based redirection, and the proposed method is compared with the DNS-based redirection method.

2.2.2.1 Domain Name System Protocol

Reaching a destination has to answer two fundamental questions. One is where to go, and other is how to get there. In CDNs, both questions are answered by using the DNS protocol [21]. DNS protocol is fundamentally developed to map human-readable form of web servers addresses to IP addresses. The Internet protocol stack consists of notations, e.g. IP addresses, that are used to direct users to servers [54]. However, such notations are not in a human readable format. At the early ages, the mapping between human-readable host names to IP addresses was dealt by sharing host-to-IP address mapping lists among the hosts. However, maintaining one master list for such mappings became hard to manage. Therefore, sharing host-to-IP address mapping lists eventually evolved to Domain Name System (DNS) [21, 55]. Thus, the Internet is crucially depending on the DNS to allow the users to map human-readable host names to IP addresses [8, 55], and to redirect the users to the best replica servers [18, 55].

The DNS protocol is the communication protocol that carries the messages between the resolvers and the DNS servers. The basics of the DNS communication protocol is defined in RFC1035 [21]. Figure 2.4 displays a map of DNS root servers placed among the countries. The DNS is a distributed and a hierarchical naming system that distributes the naming information among a large number of servers located on the Internet. When a resolver needs to translate a hostname into an IP address, the resolver soon becomes a client of the naming system and requests the host-to-IP mapping from a DNS server. The DNS server looks up its cache for the required name and replies the cosponsoring IP address to the client if available. Otherwise, the server becomes a temporary resolver of another DNS server; the server sends request messages to other DNS servers until it finds the cosponsoring IP address and returns it to the original requester. As a matter of fact, the RFC-1035 was implemented to the network simulator-3 as part of the dissertation and, the source code is publically available on Github [56].

The top level format of each message is divided into the following five sections:

1. Header: the main data structure, which distinguishes the DNS messages and the types of the DNS messages
2. Questions: a question that contains the name a resolver wanted to resolve.
3. Answer: the resource records
4. Authority: resource record that points out the authoritative DNS server (Auth. DNS server).
5. Additional: resource records that provide addition information about the query.

DNS records contain Time-To-Live (TTL) field. This field declares how long a resource record should cache before it should be deleted from the DNS servers. If the TTL is zero, that means the DNS record is valid only for the current transaction before it is discarded [21]. The TTL field is the most important field in the DNS-based CDN redirection because the TTL field verifies the granularity of adapting the content server changes. For an example, if a bad destination is chosen, it cannot be changed until the TTL expires. Therefore, the DNS-based CDN approaches could not notify users until the users contact the relevant DNS servers.

2.2.2.2 Use DNS servers to approximate the nearest service point

There are several techniques used to approximate the distance between the clients and the service point. The distance is measured regarding geographical distance, transfer delay (round trip time (RTT), the number of router hops, packet loss rate, and sometimes congestion between the service point and the users [14, 15]. However, standards are not yet published to identify the closest location by measuring the network distance or other measurements [14, 57]. Moreover, the major content providers also do not provide their approaches to determining the closest locations to the users. However, [14, 57] provided three different approaches that can be used to determine the nearest service point:

1. reactive probing
2. proactive probing
3. connection monitoring.

Table 2.2 summarizes above three methods for their pros, cons, and the limitations. In the reactive probing approach, as displayed in Fig.2.5, CDNs use a special DNS server, Reactive probing DNS server, which is commonly known as an authoritative DNS server (Auth. DNS server). Auth. DNS server, asks servers to measure the RTT for local DNS server and send the information to the Auth. DNS server. Then the Auth. DNS server selects the service point with shortest RTT [14].

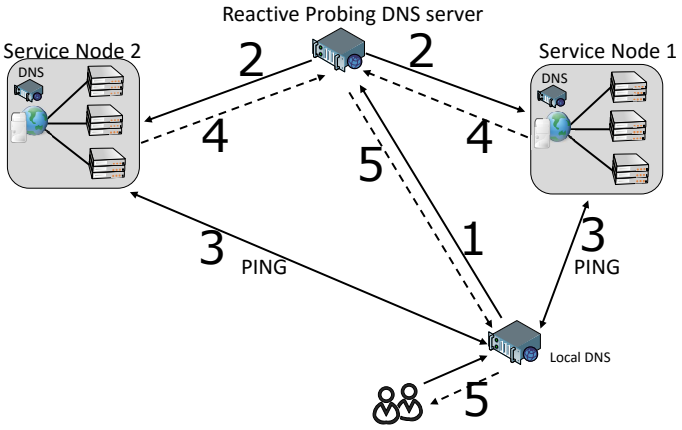


FIGURE 2.5: Reactive probing [14]
(dotted lines represent the reply messages)

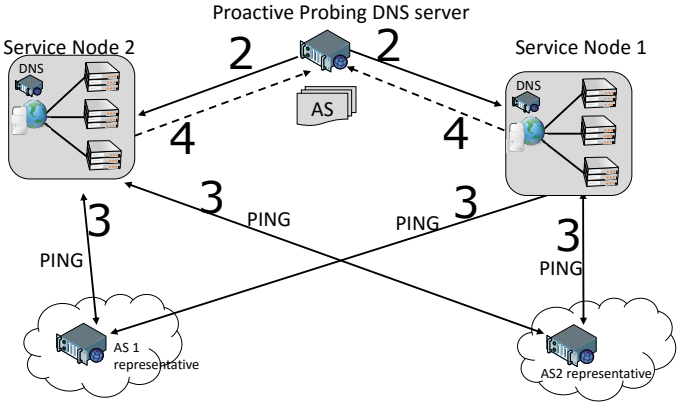


FIGURE 2.6: Proactive probing [14]
(dotted lines represent the reply messages)

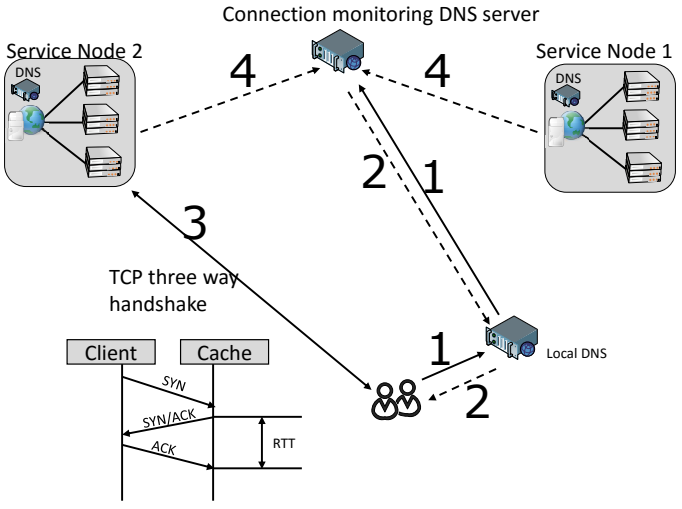


FIGURE 2.7: Connection monitoring [14]
(dotted lines represent the reply messages)

The proactive probing method also uses a special DNS server: proactive probing server, and it is also commonly identifying as Auth. DNS server. Auth. DNS server maintains a database of RTT from

TABLE 2.2: Service location approximation methods [14]

	Pros	Cons	Considerations
Reactive probing (See Fig.2.5)	- Provide the most up-to-date measurement about the distance	- Send PING messages to local DNS is not feasible - Could not adjust to the network conditions quickly - Might give false results when the network is changing	- Send PING messages to local DNS is not feasible - Could not adjust to the network conditions quickly - Might give false results when the network is changing
Proactive probing (See Fig.2.6)	- Solved the DNS response latency problem	- It is hard to find an IP address to represent an entire AS - Might give false results when approximate the IP	- Users are heavily dependent on the TTL values of the DNS to get the up-to-date server status - Servers states are not consider into calculating the distance
Connection monitoring (See Fig.2.5)	- Eliminates the probing - Measure the transport distance based on the connections - The measurement uses the state of the servers to approximate the distance.	- Assumes the local DNS and the clients are nearby - Maintain a lists of clients to select the service point - Approximate the location based on neighbors - Use the DNS in a way that its not anticipate to in its design.	- Users are heavily dependent on the TTL values of the DNS to get the up-to-date server status - Approximate TTL based on the RTT, but not based on the server state and the network state - There is no proper method to calculate TTL anticipating the state of the servers - In case of a fault no method to reduce the RTT of a DNS response once it delivers to the network.

each service point to a node list of an Autonomous Systems (ASs), representing the universe of known client networks. Figure 2.6 illustrates the scenario of the proactive probing method. The auth. DNS server sends the AS list for all service nodes and asks them to find the RTT for all nodes in the AS list. When Local DNS sends a request the Auth. DNS server provides the service point with shortest RTT for the AS that Local DNS is representing the client's local DNS server [14]. Note that reference [14]

explains all illustrated steps in Figs. 2.5, 2.6, and 2.7 in detail.

Connection monitoring method is displayed in Fig.2.7. Connection monitoring method eliminates the probing and relies on the measurement of RTT as a connection is naturally made between service nodes and the clients. This method is a comparatively accurate method that measures the transport distance from the service node to the client. The service nodes measure the RTT by measuring the delay between Synchronize/Acknowledgment (SYN/ACK) and ACK of Transmission Control Protocol (TCP) three-way handshake method (see step ❸ of Fig.2.7). Servers periodically advertise the connection monitoring Auth. DNS servers about the RTT value for each customer (see step ❹ of Fig.2.7). When local DNS contacts the Auth. DNS (see step ❶ of Fig.2.7), if the Auth. DNS server does not have a match for the particular client; the DNS server selects the servers using the basic method of DNS, using random or round robin (see step ❷ of Fig.2.7). Dissertation assumes that the algorithm is round-robin because DNS is designed to select DNS server records RR algorithm. This method might redirect the client for busy service points and clients should wait until the TTL value expires to find the next service point. One of the main drawbacks of this method is that Auth. DNS servers have to maintain a huge database to store RTT values that represent each client [14]. Moreover, the method assumes that the local DNS servers and the clients are nearby located. Moreover, this method further assumes the RTT values based on the neighbors of the data requesting client, and this system uses the DNS servers in a way that was not anticipated in its original design.

2.2.2.3 DNS-based request redirection

As explained in Section 2.2.2.2, to find the service point, first local DNS servers should contact the Auth. DNS servers. Therefore, the next question is how to find the Auth. DNS servers before finding the service point. The Lucent Technologies [58] proposes one solution, and the authors proposed product name as WebDNS. The WebDNS collects server state information from the reverse proxy servers

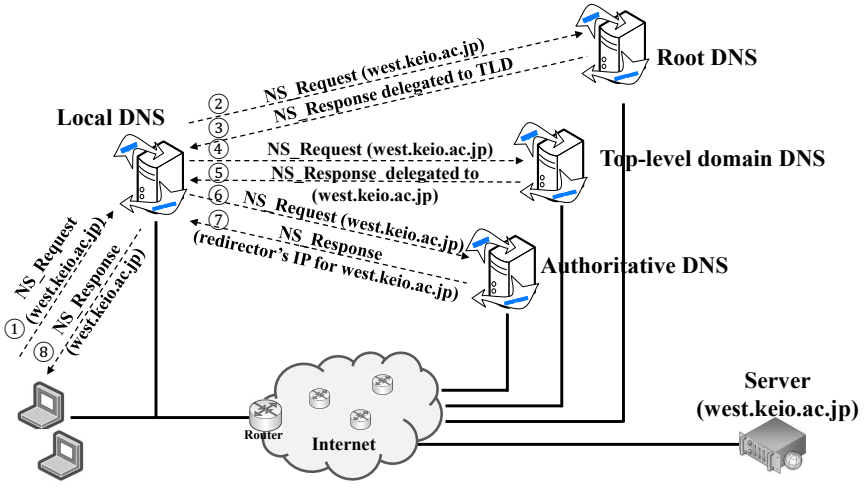


FIGURE 2.8: Recursive DNS resolution process

by using an agent based solution. The WebDNS then acts as an Auth. DNS for the content provider and they proposed a recursive hierarchical DNS resolution to reach the Auth. DNS server. Moreover, it finds the best location for a client based on the connection monitoring data. Similarly, some CDN modified the original DNS architecture and proposed new DNS architectures such as FreeFlow DNS [18, 59], to solve the service point selection problem [18, 20] (Request routing, RR). The FreeFlow DNS also proposed a similar method to go recursively through the hierarchy of DNS servers to find the Auth. DNS server. Fig.2.8 illustrates the recursive DNS resolution method.

Let us assume the scenario illustrated in Fig.2.8 for the sake of exposition. As shown in the figure, when clients request IP address of "www.abc.com" from the local DNS, the local DNS recursively queries other nameservers until it reaches the Auth. DNS server for the CDN-Domain. The Auth. DNS server selects the nearest redirector for the CDN Domain based on the location of the local DNS [14, 53]. It then returns the redirector's IP address to the local DNS. Finally, the local DNS returns the IP address to the client, and the client sends data download request (i.e., u_CDN) to the redirector. The redirector assigns the client to an appropriate surrogate server based on load balancing policies. When a given DNS strictly adheres to the calculated TTL, the local DNS servers must frequently traverse the hierarchical DNS server resolution process to inform the subscribers about the change in the servers [18, 20]. Generally, in the DNS-based redirection, the local DNS server recursively queries a hierarchy of DNS servers until it reaches the Auth. DNS server for the CDN domain [18, 20]. The caveat is the time that is spent in finding the Auth. DNS server. This factor influences the connection initiation and rapid adaptation to the changes occurring in servers.

In contrast, TTL values must be set to appropriate values so as to maintain the freshness of the redirection because downstream servers cache the DNS responses. Consequently, altered DNS architectures (e.g., FreeFlow DNS) adapt rapid changes of content servers by setting small (or possibly zero) TTL values [14, 20, 21]. However, in such methods, many DNS requests have to be transmitted in the public network rather than being served from the local DNS. Thus, Shaikh et al. also argued that small TTL values could significantly degrade the scalability of the DNS [53]. Moreover, the hierarchical nature of the DNS resolution process increases the nameserver resolution latency as well. Furthermore, CDNs redirect their subscribers without being acutely aware of the ISP network conditions [27].

2.2.3 Routers, then and now

Traditionally, routers have been implemented purely with software running on a general-purpose personal computer with some interfaces. Such devices can receive packets on one interface, i.e., ingress interface, and forwards the packets to the other interface towards the destination of the packet. Eventually, with the development of silicon technology, router vendors have made the PC routers to hardware-based routers, which can handle high data rates. Broadly speaking, a router must perform two fundamental tasks: routing and packet forwarding. Routers construct a view of the network topology and

compute the best path based on the information exchanged among the neighbors. The network topology reflects the network destinations that can be reached from the routers through IP prefix-based network address blacks.

When considering the basic forwarding functions of the routers, the forwarding function is responsible for following actions:

1. Gather network information and maintain the topology table according to the network conditions
2. Validate header structures and update the header structures as necessary.
3. Lookup route table and forward the packet to its next hop towards the destination.
4. Handle some basic service controlling options such as QoS and use ICMP messages to notify about the errors.

Besides the basic roles, the market has urged the need for an additional complex function to the routers. That is, with the popularity of the Internet, complex issues such as security, user requirements, and service guarantees based on different service level agreements have become important and routers were developed to address such features. This complex packet processing incurs additional processing and memory for the routers. Consequently, routes are developed to provide a variety of mechanisms such as follows:

1. Packet classification: For distinguishing packets, a router might need to examine not only the destination address but also source address, source port, destination ports, and the application layer information. The process of differentiating the packets and applying the necessary actions according to the certain rules is known as packet classification.
2. Packet Translation: Packet translations are needed to execute address mappings according to the application requirement. For example, as the IPv4 address space is exhausting, there is a need to map several host addresses to a single public address. Such features required routers to maintain a certain level of storage modules.
3. Traffic prioritization: Traffic prioritization techniques are required to guarantee a certain quality of service agreements with the service providers and the ISPs. For example, a certain agreement might say that fixed amount of packet should be transmitted at a given time interval, which is an essential feature for real-time streaming multimedia applications.
4. Routing policies: Routing policies are necessary parts of the routers to maintain the user forwarding policies according to the agreements with the ISP and the business bodies. Specifically, when CDNs increase the number of servers in their data centers, the routing policies are applied to redirect the traffic necessitated by the CDN.

TABLE 2.3: Novel high end routers

Vender	Router name	Interface details	Expected throughput	Year
Juniper	T160	40Gb Eth x 16 Or 100 GB Eth x 8	1.6 Tbps	2007
Juniper	T4000	40Gb Eth x 16 Or 100 GB Eth x 16	4Tbps	2011
Cisco	CRS-3	10Gb Eth x 320 Or 100 GB Eth x 16	4.48Tbps	2010
Alaxala	AX8632R	10Gb Eth x 192 Or 100 GB Eth x 16	6.4Tbps	2013
Cisco	16-Slot	400Gb Eth 16	12.8Tbps	2014

Such requirements urge lots of processing power and memory in the router. Furthermore, router manufacturers such as Cisco, Juniper, Hitachi, and Motorola are proposing to use their routers to provide user-based services by placing the routers at the edge of the ISP networks [29]. Eventually, routers urge special packets processing units such as cache processors and multicore processors, network adapters, RAM, and storage modules to provide effective users services.

Lately, major router manufacturers, e.g., Cisco, Juniper, and Alaxala, have recently revealed programming interfaces that allow packet manipulation by third-party applications and the addition of new services to routers [60, 61], i.e., process Internet of Things data at the edge routers [29]. Similarly, in [62], Kim et al. proposed to implement a high-performance router by shrinking a rack of computers into a single hardware box. Further, Intel introduced a packet processing framework, Intel data plane development kit (DPDK) [63], to analyze packets and simultaneously maintain high throughput packet flow; Intel claimed that they could analyze packets by providing a throughput of up to 160 Gbps [64]. Studies [65, 66] projected such trends and used cached content on routers for future CDN implementations. Eventually, the routers are again going towards the PC passed high-performance, high throughput routers. Table 2.3 provides a basic comparison about high-end novel routers.

Recently, Cisco proposes to provide users services such as WAN acceleration [67], Session Border controlling [32], Ad incursion [30], Firewall services [30], load balancing [30] by placing high-end routers at the edge of the network, i.e., as gateway routers. Moreover, Hitachi is also proposing WAN accelerator [33] and place at the WAN accelerators at the edges of the network to accelerate the TCP connection by intercepting them in the middle of the transmission. This technology is essential to encase the content navigation in CDNs. Furthermore, Cisco is introducing their Fog computing [68] architecture to process Internet of Everything (IoE) [69] at the edges of the network. Cisco is not alone; Qualcomm has a similar vision of router architecture that has the capability of content caching, and a computer sets to manage a house full of connected devices, including running algorithms that govern which devices can talk to and when they might have access to the internet [29]. Successively, with the development of such technologies, Akamai and Qualcomm think one solution is to cache popular digital content in the home or office. In 2014, at the Consumer Electronics Show (CES),

Qualcomm's Atheros showed off its IPQ "smart gateway", a router device running Akamai software which "prepositions" bandwidth-heavy content [70].

2.3 Network path selection

When content goes from source to its intended destinations, the content should navigate through a network [71]. A network is an entity, which consists of routers and links between the routers. In simple terms, a packet that carries the content should travel through several cross-points, and those cross-points are known as routers. As previously described, there are two basic functions that a router should ensure; 1) maintain an information base which consists of the destination networks and the corresponding neighbors to reach those destinations, and, 2.) select network paths and forward the packets towards their destinations. Network path selection defines the methodology of selecting the best routes and forwarding traffic in a network. Networks use routing protocols to select network paths to provide uninterrupted Internet access to their subscribers. In general, routing can be summarized as the process that routers decide how to forward a datagram based on its destination address by comparing the network states information that router keeps in a special entity known as a forwarding information base, i.e., routing tables [72]. Routing tables contain entries for each network the router can reach, what is the adjacent node the router should contact to reach each destination network, the cost of reaching the next hop, and the validity of the next hop. Apparently, the routing tables are vital to the network path selection process because routing tables provide the status of the network [8, 41, 71, 72].

There are two types of routing protocols available for network path selection. The first category is interior gateway routing protocols (IGPs). IGPs are used to exchange the route information among the routers within an autonomous system (AS) [72]. The other type is Exterior gateway routing protocols (EGPs). The EGPs are used to exchange the route information among several autonomous systems.

TABLE 2.4: Comparisons of IGPs [8]

Name	Type	Updates	Metric	Parameters used for calculate the metric
RIP	Distance vector	30s	Hops	Number of hops to the destination
RIPv2	Distance vector	30s	Hops	Number of host to the destination
IGRP	Distance vector	90s	Composite metric	Combinations of bandwidth, delay, load, and reliability
EIGRP	Hybrid	Link state updates	Composite metric	Combinations of bandwidth, delay, load, and reliability
OSPF	Link state	Link state updates	Cost	Calculated according to the link bandwidth
ISIS	Link state	Link state updates	Cost	Calculated according to the link bandwidth

Given that dissertation is only considering about ISPs, and ISPs can be interpreted as an AS, the dissertation focuses only on IGPs. The IGPs can be categorized into three different types:

1. Link state protocols,
2. Distance vector protocols
3. Hybrid protocols (also known as advanced distance vector routing).

An overview comparison of IGPs is given in Table 2.4. As stated in [49, 73] and presented in Table 2.4, current IGPs, i.e., "shortest path routing protocols," are optimized for a single arbitrary metric, administrative weight. However, IGPs lack the ability to consider the impact of network devices for network path selection, e.g., the impact of a router. Hence, as stated previously routers, i.e., network middleboxes, are becoming pervasive on the Internet for providing efficient end-user services [8, 41, 71, 74]. When traversing links with such middleboxes, it is important for IGPs to consider the impact of middleboxes for network path selection.

Furthermore, as discussed early in this chapter, CP networks are deployed within the ISP networks, e.g., CDN [73]. In such networks, CPs maintain distributed server placement structures and redirect their subscribers among the content servers according to either proximity or end-to-end latency [49, 73, 75]. Therefore, in such networks, primary assumptions of network path selection, i.e., the traffic matrix is point-to-point and constant, are violated [27, 73]. Moreover, as stated in [73], when an ISP optimizes its IGP for a particular CP, the IGPs lack the ability to consider non-CP traffic for network path selection. Thus, it is important for IGPs to contemplate the dynamic behavior of network devices for proper network path selection because such network devices reflect the dynamicity of the traffic matrix introduced by the current Internet architecture [73].

Given that a single IGP does not necessarily consider the dynamic behavior of network devices for network path selection, such protocols introduce a traffic engineering (TE) extension to monitor extended link attributes [76, 77]. In practice, ISPs use such TE extensions conjunction with label switching protocols [78] to use the extended link attributes for operative network path selection [41, 79, 80]. However, as stated in [73, 81, 82], for certain traffic patterns, such approaches overload some links in the network while leaving other links unutilized at the same time. Moreover, providers incur configuration complexities, interoperability, and compatibility issues of proprietary protocols and devices that lead to sub-optimal network resource utilization [83, 84]. Thus, ISPs require a general IGP to solve the route optimization problem [82]. Aside from such common practices, as stated in [79, 85], ISPs may create complex, yet effective, systems by combining several protocols: information gathering [76, 77], resource reservation [86], routing [8, 41, 71], and label switching [79] for efficient network path selection. Despite the advantages, because of 1) interoperability issues among protocols and devices [83, 84], 2) the cost and link usage for protocol management, and 3) high implementation time, most ISPs are yet to implement such systems, leaving behavioral consequences to be explored thus far.

2.4 Related works

The necessity for cooperation between network and content providers has been a topic of discussion within the research community and among vendors. In [87], Xie et al. proposed a communication portal between ISPs and P2P applications; this portal can be used by P2P applications to determine ISP-based network information to reduce the cost for network providers without sacrificing performance. Similarly, in [88], Aggarwal et al. proposed an Oracle service run by the ISP; the users can use the ranked neighbor to improve the performance metrics. More specifically, in [27], Poese et al. proposed ISP-CDN collaboration. The authors used the traffic matrix of the CDN for TE, and hence, proposed to redirect end-users to the nearest content servers in small time scales.

Also, some researchers proposed a method to optimize the network path selection using different criteria. In [82], Antic et al. pointed out that the shortest path routing (SPR) protocols used on the Internet today do not maximize the guaranteed node traffic loads, and do not provide scalable and fast bandwidth reservations. So the authors proposed that load balancing can improve the network throughput for arbitrary traffic patterns that are generated due to the dynamic content navigation. Consequently, in [82] the authors analyzed and implemented a routing protocol that is based on load balancing and a commonly used shortest path routing protocol and is, consequently, termed as LB-SPR.

There are several proposals on introducing network imtermediaries and middlewares into the ISP for collaborations. For example, in [89], Raghavan and Snoeren proposed present "Platypus," which is an authenticated source routing system built using the concept of network capabilities. As the authors anticipated, network capabilities allow for accountable, fine-grained path selection by cryptographically attesting to policy compliance at each hop along a source route. Moreover, the capabilities can be composed to construct routes through multiple ASes and can be delegated to third parties. Even though, authors were interested in implementing a system for routing, the authors do not consider the states of the network devices to build the routing policies alongside with the capabilities. Hence, this approach is an important asset for the foreseeable future upgrades of the study conducted in the dissertation; future visions are explained in Chapter 6 Section 2.

Moreover, various studies have attempted to learn the confronted content navigation problem due to the emerging current Internet architecture. Among them, [49, 74] state that CPs should use the ISP route matrix for effective end-user redirection. The authors of [27, 73] stated that ISPs should use the dynamicity of the CP traffic matrix for network path selection; the metric for network path selection should be a function of the network state information instead of an IGP, which is optimized to a single metric. Also, some studies, such as [82, 90], have attempted to optimize the existing IGPs for effective network path selection according to the prevailing traffic matrix. Alternatively, studies such as [85, 91] have created complex systems to consider network state information for content navigation. However, in [82], the authors argued that the NSPs require simple, yet effective, IGPs for network path selection according to the state of the network, which is the main motivation of ESLR.

When considering the dynamic routing paradigms, in particular, delay-based routing, the old ARPANET routing protocol used link propagation delay for network path selection [92]. However, the protocol showed a substantial amount of route oscillations because the link propagation delay was averaged over the ten-second interval. Later, Khanna et al. revised the ARPANET protocol and devised a new concept for calculating a delay-based metric under light link loads and a capacity-based metric under heavy link loads [92]. However, the method used to convert "delay" into "capacity" is unclear, and the authors were not interested in using state information of routers as a parameter for metric calculation. Furthermore, the proposed method can only be applied to networks that consist of several small node-to-node flows. Another delay-based routing protocol is presented in [93]. The authors proposed a new method for using delay as the metric for the Babel routing protocol for data center networks. The authors claimed that their approach could successfully compensate the route oscillation problem of data center routing protocols. Unfortunately, the authors were not interested in using the effect of routers for their route metric calculation. Furthermore, the proposed approach measured the route metric, RTT, by sending "hello" messages to the nodes in the topology. This might increase network loads and the number of protocol messages in the network.

In [26], Valancius et al. proposed a joint routing approach to effectively find the network paths between clients and the replica. Authors have proposed a system to select both network path selection and content server selection by placing a virtual node in between the clients and the actual service locations. Then the authors optimized used the virtual replica servers to identify both network and server performance by analyzing the conditions of both domains. Then the authors have redirected the users to the replica server using by creating a metric using both network and content server conditions. The authors were concentrated on a global level content server placement, and this might create a problem in deploying their proposed system into an existing online service provider network.

The study that is most similar to the dissertation is [49] in terms of proposing a collaborative infrastructure to improve the content delivery. In [49], the authors selected a server by utilizing network views collected from an ISP; thus, the end-users were redirected to the most efficient server instead of the nearest server. The authors revealed that the ISP-CDN collaboration could be implemented in real ISP networks by simulating their system in the Tier-1 ISP. However, the authors were not interested in using the content server information for network path selection. Further, their proposal is considerably dependent on hierarchical DNS and does not discuss the problems of latency and the lag in adapting to the server changes due to hierarchical DNS resolution.

In [65], the ISP-centric content delivery (iCODE) architecture was studied; it was designed to cache the content in intermediate routers in the ISP network. iCODE assumed routers would have in-network storage modules, which are similar to SoRs. iCODE used the swarming technique to download the data from routers to end-users. However, the authors were only interested in caching content on the routers. They do not consider the topic of content-based services by assuming more intelligent routers in their system.

In [94], the Data-oriented network architecture (DONA) was proposed. DONA resolves the content by using flat and self-certifying names instead of URLs. Hence, in the DONA architecture, DNS name resolution is replaced by a name-based anycast. In this architecture, name resolution is performed by using a network entity called resolution handler (RH). To use DONA in a network, the ISP must replace the DNS servers by RHs. Although the DONA architecture guarantees the perfect global availability of content, the system functions only when RHs are deployed on all ISPs.

Chapter 3

Software Service-oriented Router

This chapter discusses the proposed software Service-oriented Router (SoR) design and implementation; the software SoR is one of the main three objectives of the dissertation. First discussion of SoR was started in mid-2008 by proposing the semantic router; semantic router was proposed to use data streams to enrich user services [34, 35]. In fact, the motivation of the semantic router was to design a service-friendly router that provides unique content-based services to enhance end user services. Moreover, the first glimpse of the SoR discussed a novel query language towards storing data in network routers. In essence, the SoR can be introduced as a router with an ample storage module and packet processing cores to capture packets, analyze packets for valuable information, store necessary information, and to use that information for enhancing user services. This idea has the potential to shift current the IP-based Internet architecture to a service-based open innovation platform.

As illustrated in Fig.3.1 the SoR is a collaborative attempt of developing a complete router, which provides user services from the router itself. Figure 3.1 elaborates the work segregation of the SoR,

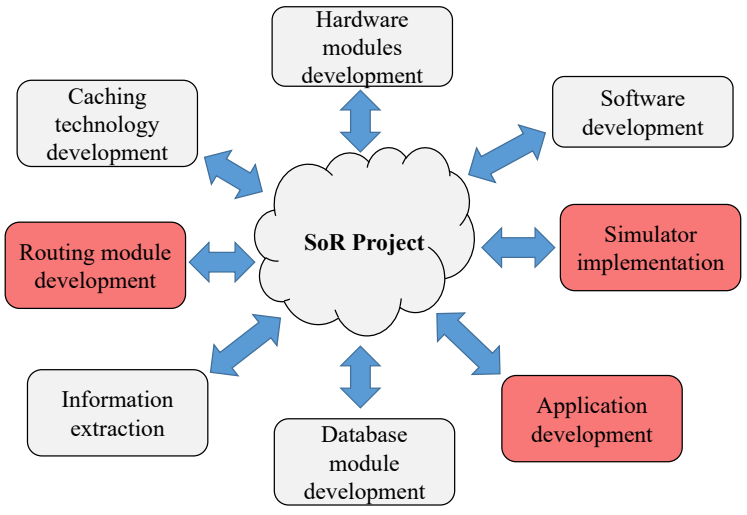


FIGURE 3.1: Service-oriented Router project

achieving a router that has the capability of providing services as predicted by Inoue et.al in [34, 35]. When considering a CPU-based general router architectures [95], routers contain packet processing units, memory management units, communication path optimization units, and caching mechanisms. However, when developing a router with the capability of providing user services, i.e., SoR, there is an immense endeavor because, in addition to the components mentioned above, the router should focuses on application layer packet capturing, analyzing, storing, and providing user services. Namely, The SoR project focused on routing module development [96, 97], packet capturing and analysis at wire rate [98], store the captured data into databases and maintain the required throughput [99, 100], support required caching mechanisms [101], data path acceleration and modifications, application development [102, 103], and simulation development to check the effectiveness of proposed applications [104].

3.1 Overview of general Service-oriented Router

Figure 3.2 elaborates the latest version of the SoR being developed [34, 35, 96, 98–103] when the dissertation is being drafted. As illustrated in the figure, in addition to the function that a general router provides, the SoR is mainly focusing on capturing and analyzing application layer, i.e., L7, data and storing those data in high-throughput databases in order to provide foreseeable user services. The high-level functions description is as follows:

1. Initially, SoRs parse the data as required by the operator. At this stage, SoRs copy the data for further analysis on necessary information.
2. In the analysis phase, SoR reconstructs the data streams according to the five tuples: Source-IP, Source-Port, Destination-IP, Destination-Port, and Protocol number.
3. Then, the data are checked for further encodings such as chunk encoding and GZip compression, and handle them using necessary techniques.

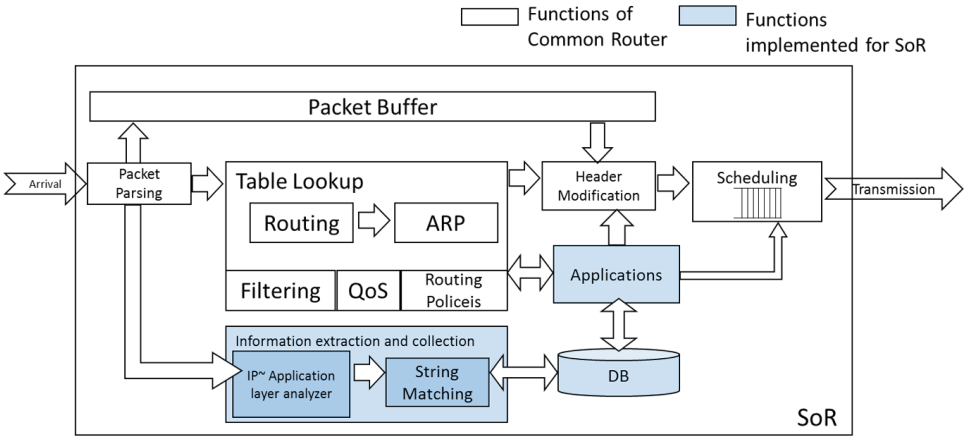


FIGURE 3.2: Overview of SoR

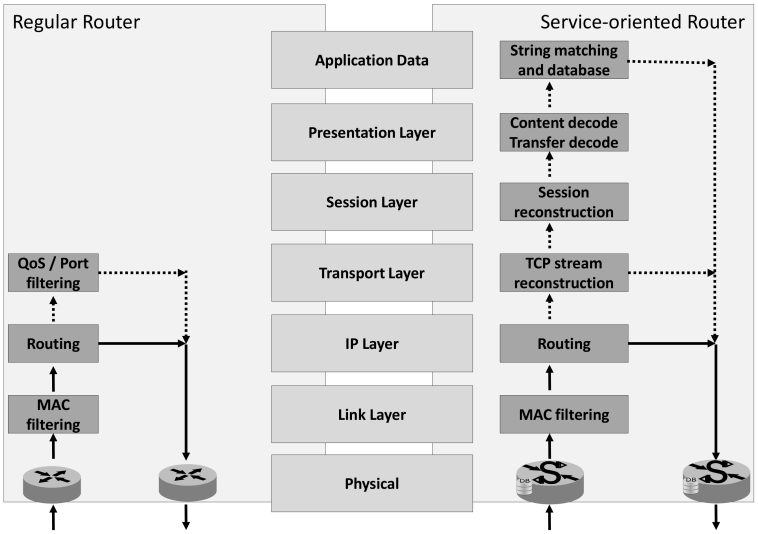


FIGURE 3.3: SoR Vs. a regular router

4. After the data have been extracted and arranged according to the user sessions, SoRs match the required data according to some configured rule set using string-matching techniques.

5. Finally, SoRs store the data in high-throughput databases.

Consequently, the SoR architecture is being developed to optimizes the packet capturing, string matching and database insertions operations using both hardware and software techniques [14]. As illustrated in the Fig.3.3, the SoR is working across all the layer of ISO/OSI reference model, while regular routers access information up to network layer. This means that SoR requires sophisticated techniques to minimize the delay introduced by analyzing the data up to application layer information. Following sections provide brief explanations and the necessary references to understand the technology behind the main components of SoRs.

3.1.1 Packet capturing and analysis

This module is the part of "information extraction and collection module" (See Fig.3.2), and this module is a highly sophisticated module, which requires the superlative optimization to minimize the delay caused by SoRs, providing application layer data extraction before packet routing. Furthermore, the packet capturing module should be capable of handling all necessary protocol constraints to make sure that the packets are getting not impaired by sniffing in the middle of packet routing. Given that web services are increasingly attractive because of the rich content provided by flexible and powerful APIs when network applications exchange large data over the Internet, the data is divided into several or many packets. For example, a bundle of TCP/IP packets is known as a TCP streams. If the routers are capable of DPI such streams and extract the information, the information can be used by the service providers to provide user services effectively, e.g., solve CDN content navigation problem. Moreover,

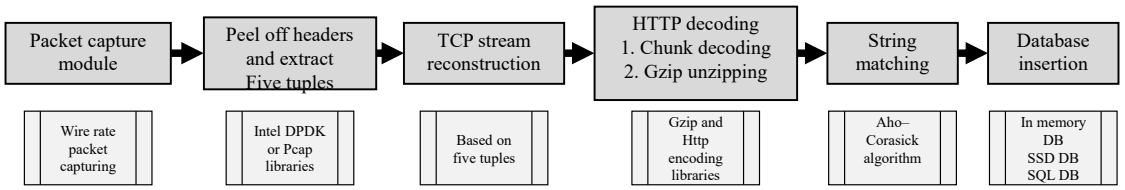


FIGURE 3.4: Functional diagram of information extraction process of SoR

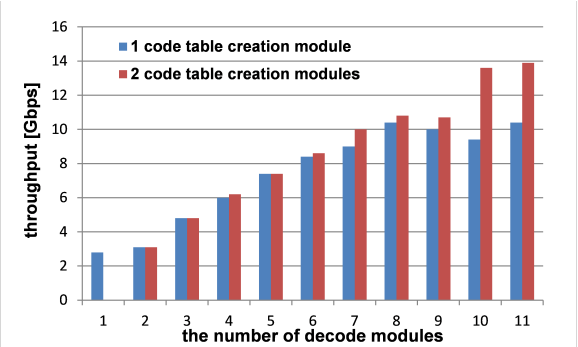


FIGURE 3.5: Relation between throughput and number of modules in parallel processing [106]

the benefit of handling the content of packets at a router is great because the routers are well placed in a network infrastructure to reinforce both routing and application layer policies. However, routers are not widely employed by existing network infrastructures to provide such user services.

HTTP is the widely used protocol on the Internet, and it is a communication protocol, which packetizes HTML packets, multimedia content, text, and hyperlinks, used to maintain a communication channel between web servers and clients. As a matter of fact, recent traffic analysis studies are pointing out that more than 60% of the current network traffic is HTTP [105]. So that the DPI modules should be optimized to extract information from the HTTP streams without slowing down the packet transfer speed. Furthermore, routers require capturing data at the links and pass them to necessary applications that are capable of analyzing the data for required information. There, routers prerequisite libraries such as Pcap [19], and such libraries have bottlenecks of copying packets from the hardware layer, network interface cards, to the application layer programs. The reason is the memory management. Consequently, Intel Introduced a novel framework, Intel data plane development kit (DPDK) [63], to copy data with zero process cycles using their hardware and programming libraries. Intel claims that they can copy data and analyze data by maintaining 160Gbps throughput [63]. Consequently, the latest packet capturing modules developed into the SoR uses DPDK and the results show that the proposed application can capture, analyze, and route data by maintaining 40Gbps throughput.

When considering the packet information extraction phase, data extraction from HTTP packets is crucial. In HTTP / 1.1 or later, most of the data are initially compressed by content encoding, i.e., gzip, and after that, the resultant bytes are encoded by transfer encoding, i.e., chunked encoding. Consequently, it is impossible to analyze the compressed packet up to application layer information without

TABLE 3.1: System configuration parameters [107]

Software	**Details**	
OS	CentOS 6.3 with, kernel ver. 2.6.32-279.22.1.el6.	
Rule set	Snort Rules v2.9	
CUDA	CUDA toolkit Ver.,4.2	
Compiler	NVCC compiler ver.,4.2	
Component		**Details**
Host	CPU	Intel Core i7-3930K,CPU @ 3.2 GHz
	Memory	32GB DDR3 @ 1,600 MHz
Device	GPU	NVIDIA GeForce GTX 680
	Memory	2,048MB 256bit-GDDR5 memory

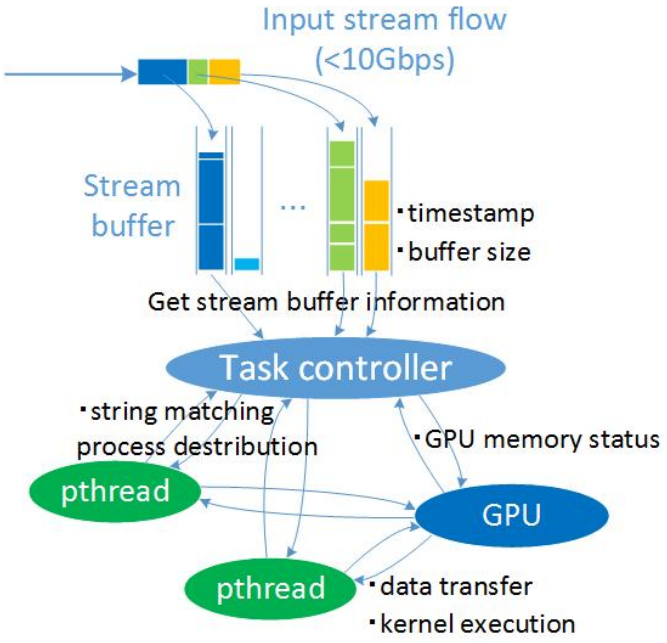


FIGURE 3.6: Implementation of process distribution and monitoring mechanism [107]

decompression and decoding. Therefore, as presented in Fig.3.4, the SoR follows up sophisticated procedures to analyze the application layer information by decoding and unzipping the data. SoRs needs to collect all packets for each stream and decode them individually. Such processes consume much memory and processing power, resulting slow down the packet propagation. Nonetheless, [106] incurs that almost all studies, which contemplates DPI on routers, do not consider DPI up to application layer by decompressing the HTTP because it slows down the packet propagation. Moreover, since it is hard to achieve a high-speed network processing more than 10Gbps using a GZIP decoding mechanism, there was no suggestion about architecture, which is implementable on the router. Thus, [106] proposes an architecture to decode HTTP compression data in wire-rate on a router in a 40Gbps networks, and the authors pointed out that they can maintain more than 10Gbps throughput (see Fig.3.5).

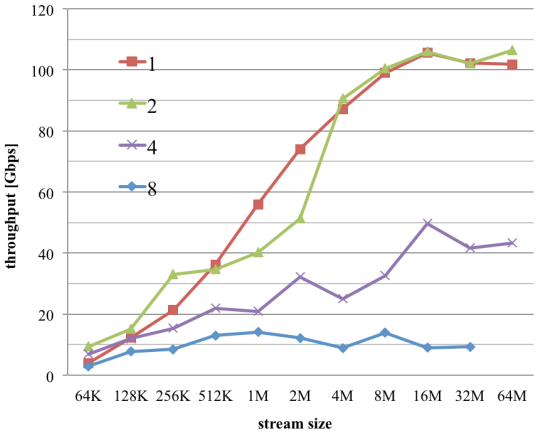


FIGURE 3.7: Throughput comparison [107]

3.1.2 String matching

As illustrated in Fig.3.4 when the data are extracted from the packet streams, the SoRs needed several techniques using the extracted information to provide user services. For an example, if we consider an intrusion detection system, SoRs should search the application data with malicious signatures to detect the spoiled packets. When SoR detects malicious packet streams, it suspends the forwarding process of the malicious stream, and hence, routers can notify the clients that they are a target of an attack. If we consider CDN networks, the HTTP Get packets contain the URI, which indicates the content that the user wanted to fetch on. Therefore, when SoRs receives such messages, the SoRs should analyze the URI/URL to detect the content for providing necessary services such as solving "service point selection" of "user navigation problem." In simple terms, such kind of high-level functions requires string-matching capability on SoRs.

Consequently, SoRs use string-matching algorithms for searching multiple text patterns for received data. String matching algorithms can be classified into two types, 1) string matching for a single pattern, 2) string matching for a set of patterns consist of multiple patterns simultaneously. Hence, SoRs use the Aho-Corasick algorithm (AC algorithm) [108] to provide string matching. This algorithm executes on both general purpose graphics processing units (GPGPUs) and Intel i7 Processor to extract the information from traffic streams and match the strings using preconfigured string formats. The string formats are assumed to the SNORT rules presented in [109]. Figure 3.6 shows the implemented string matching architecture using NVIDIA GeForce GTX 680 GPGPU. Moreover, Table 3.1 provides the configuration parameters were used to yield the results given in Fig.3.7. According to the results, the string matching process, for the stream buffer that stores more than 16MB data, achieved up to 108Gbps throughput, and that is 3.4 times higher than the performance achieved [110].

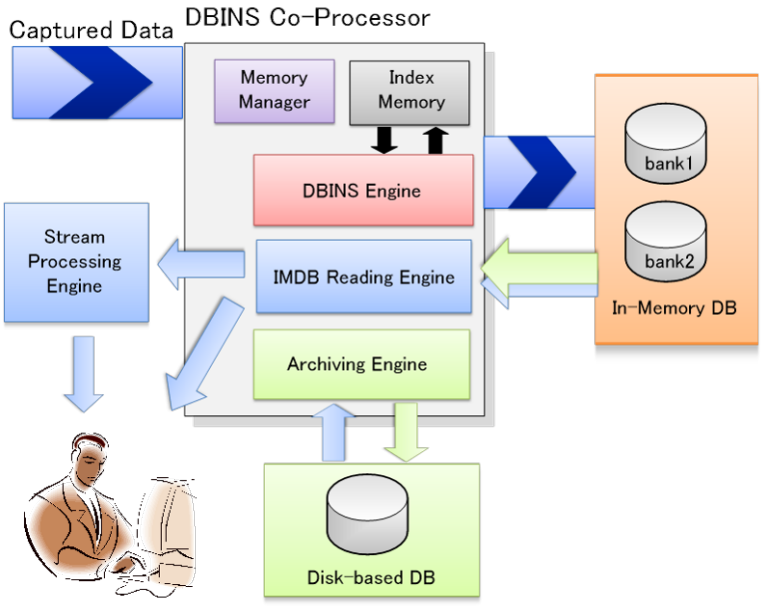


FIGURE 3.8: DBINS Co-processor [99]

TABLE 3.2: Keys and pointers used for indexing [99]

Keys and pointers	Contents
ID	Leaf-ID
Timestamp	Packet arrival time
Packet Size	Packet Size
Share Number	The number of owners of captured data
Prev. T-Pointer	Previous pointer of Time-list
Next. T-Pointer	Next pointer of Time-list
Next. ID-Pointer	Next pointer of ID-list

3.1.3 Database insertion

As stated earlier, the SoR can be inferred as a router with an ample storage module and packet processors to provide user services. However, when considering the current Internet architecture, there is an enormous amount of network traffic flows in a gigabit throughput. This means that the SoRs have to capture network traffic selectively and store them in databases carefully. Moreover, to prevent outflow during capture, high-throughput data insertion functions are mandatory compared to the querying databases. Certainly, this is a requirement of the application to which the users are requesting. Thus, it is essential to maintain a throughput of database insertion more than 10 Gbps over a core, metro, and edge networks, whereas to maintain 1 Gbps in a local area network such as business enterprise. The SoR supports both hard disk based Databases as well and in-memory databases (IMDB). The selection of the database and the traffic patterns are heavily contingent of the application requirement.

As presented in Fig.3.8, Nishida et al. [99] present an IMBD engine for the SoR when this dissertation been drafted. As shown in the Fig.3.8, a Database Insertion Co-processor was proposed as a

core function of the database insertion hardware. The DBINS Co-processor consists of three hardware modules; DB Insertion Engine (DBINS Engine), Archiving Engine, and In-memory Reading Engine. The DBINS Engine allows inserting captured data into IMDB at a multi-gigabit throughput rate. The Archiving Engine migrates data from IMDB to a disk-based database while the IMDB Reading Engine is a mechanism for preprocessing the essential functions of stream processing such as selection, projection and joining. In [99], the DBINS Engine is focused on satisfying wire-rate database insertion.

When DBINS Engine stores network stream into an IMDB, it is not a practical approach to storing all data due to the cost of hardware and software resources, and data insertion might drop an enormous amount of network traffic. Therefore, it is necessary to extract the required data selectively according to the necessity of the application. It is also desirable that the data be listed based on the issued query, where the stored data can be loaded as a series of data by high-speed selection processing. Therefore, the DBINS Engine creates an index that is accessible to the stored data according to Leaf-ID, which is the ID query key. Subsequently, the DBINS Engine creates an index as shown in Table 3.2 [99]. Note that; the DBINS Engine manages the index of the captured data, so it is scalable with an increase in queries because the resources of the DBINS Co-processor are limited. The DBINS Engine was implemented using Verilog HDL by synthesizing it on an FPGA (Vertex5 XC5VLX330T) and further implemented it with FREEPDK45n Technology as ASIC (Application Specific Integrated Circuit). The test results showed that the available throughput was 8.25 Gbps with a 50-byte packet size and 17.1 Gbps with 1306-byte packets when the DBINS Engine was synthesized on FPGA. Moreover, the throughput was 27.0 Gbps with a 50-byte packet size and 55.9 Gbps with a 1306-byte packets size, when the DBINS Engine was synthesized in ASIC.

3.1.4 Applications developed using SoR

Since 2008, when the first idea of the semantic router was presented, the SoR, as a whole or partially, used for several applications. Among them, packet capture and analyzer implementation by Ishida et al., website recommendation by Masuda et al. [103], web page data extraction by Takagiwa et al. [111], CDN load balancing by Erwin et al. [112], privacy preserving application by Sawada et al. [113], secure routing protocol by Tennekoon et al. [114], and enabling service-oriented communication platform for smart cities by Kubo et al. [115] are highlighted studies. Table 3.3 summarizes the study area, how the SoR is used, the primary purpose of the study, and a brief introduction to that study.

TABLE 3.3: Studies conducted by using the SoR

Study title	Author	How SoR is used	Summery
Hardware acceleration and data-utility improvement for low-latency privacy preserving mechanisms	Sawada et al. [113]	A hardware acceleration for SoR to preserve user privacy	With the growth of the internet, users are required to share their private and sensitive information. However, users value privacy when sharing such sensitive information. This study was proposed to preserve user privacy using k-anonymity and l-diversity method using hardware acceleration.
Per Hop Data Encryption Protocol for Transmission of Motion Control Data Over Public Networks	Tennekoon et al. [114]	A proposal to use SoR to route bilateral control data in a secured environment.	Authors proposed a per-hop data encryption mechanism using SoRs. The network infrastructure was tested to send bilateral control data, which required a secure environment to send data.
Service-oriented Communication Platform for Scalable Smart Community Applications	Kubo et al. [115]	Use the concept of SoR to build future service-oriented communication platform.	The authors were interested in using the service-oriented router based communication networks to enable future service-oriented communication platform. The proposed platform successfully enhanced the building energy management system successfully in urban areas.
An Efficient Technique of Information Extraction Processing in Packet Data Management Infrastructure	Ishida et al. [98]	Capture and extract the packet information	A novel method that was proposed to extract the valuable information required to provide user services services. This study measured and evaluated the necessary resources for both hardware and software to extract information at wire rate using a software simulator named as SRIM.

Continued on next page

Table 3.3 – continued from previous page

Study title	Author	How SoR is used	Summery
Website-transverse Recommendation application based on Content captured by Network Router	Masuda et al. [103]	Capture and recommend the pdf files based on the user interest on the data	The SoR has been used to provide a recommendation service based on the information such as *Who did what, when and where*. Authors proposed to recommend pdf files for the academic institutes base on the users' interest on Internet browsing.
Local Trend Detection from Network Traffic Using a Topic Model and Network Router	Takagiwa et al. [111]	Capture traffic and detect user trends by placing SoRs at the edge of the network	This study is a trend detection application based on network traffic. Authors proposed to identify the URLs by performing DPI over Internet traffic streams and eliminate of the non-relevant traffic such as advertisements, some CSS and JavaScript.
Modeling of Router - based Request Redirection for Content Distribution Network	Erwin et al. [112]	Use SoR to provide CDN load balancing.	Proposed to use SoR as load balancers of the CDN networks. Authors proposed a mathematical model to place SoRs in CDN and use the SoRs to redirect end-users based on the load of the servers.

Consequently, as presented in Table 3.3, the SoR has been used for many application in a range of general Internet communication to send controlling data such as bilateral data. Moreover, the SoR is proven to use as a high-performance device, which can be used to enhance the user applications effectively. However, when the study of the dissertation has been carried out, the primary functions of the SoR such as DPI, in network databases, security, and privacy preserving features, were partly used by each author to evaluate their research themes accordingly. Therefore, as pointed out in Chapter 1, a simulator to simulate general SoR functions is required as one objective of this dissertation. Consequently, dissertation proposes to implement a software SoR using two well-known network simulators,

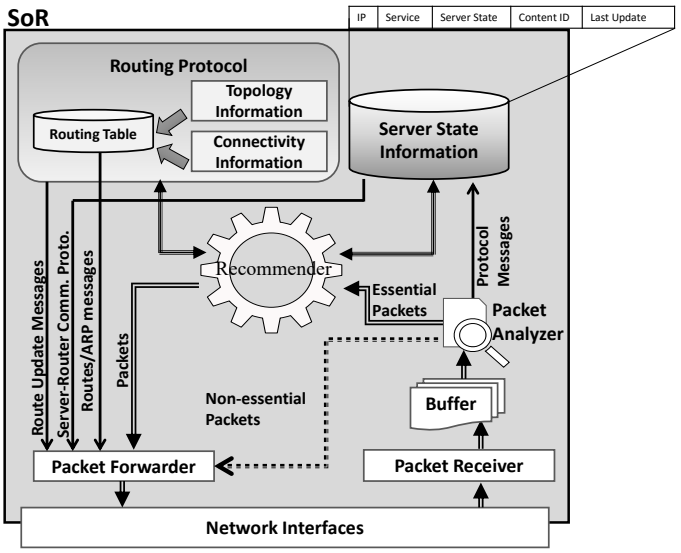


FIGURE 3.9: Proposed software SoR

ns-2 [116] and ns-3 [38, 104]. And implemented the general SoR into both those simulators and tested for CDN applications. The Software SoR implementation is described in the following section.

3.2 Proposed Software Service-oriented Router

The service-oriented Router presented in Section 3.1 continues to be a topic of research and development at the stage of this proposal work had been carried out. Members of several laboratories, research institutes, professional bodies, and industrial experts are joining to develop the SoR as a complete router with the motivation of shifting the current Internet architecture towards the service-based open innovation platform. Software and hardware components of the SoR are currently being drawn up to create an entire router since the motivation of the SoR project is to develop and commercialize the SoR within several years. In fact, the modules of the SoR was segregated among researchers and they are developing those modules in parallel (see Fig.3.1). Hence, there was no a hardware device, i.e., a router, known as a SoR at the time this dissertation was carried out. Therefore, as one of the three objectives of the dissertation, a software SoR is proposed using two well know network simulators (ns-2 and ns-3) to simulate possible application developments.

The dissertation proposes implementing a simulator to emulate software SoR because, in the field of research and development, simulations, and the simulators play a significant role towards the development of protocols, devices, theories, applications, and ideas. Consequently, to develop the software SoR with prevailing standards, a general and well-known simulator was mandatory. Among many simulators such as OPNET [117], NetSim [118], CORE [119], and GNS3 [120], both network simulator-2 (ns-2) [116] and, its successor, network simulator-3 (ns-3) [121] were selected to develop the software

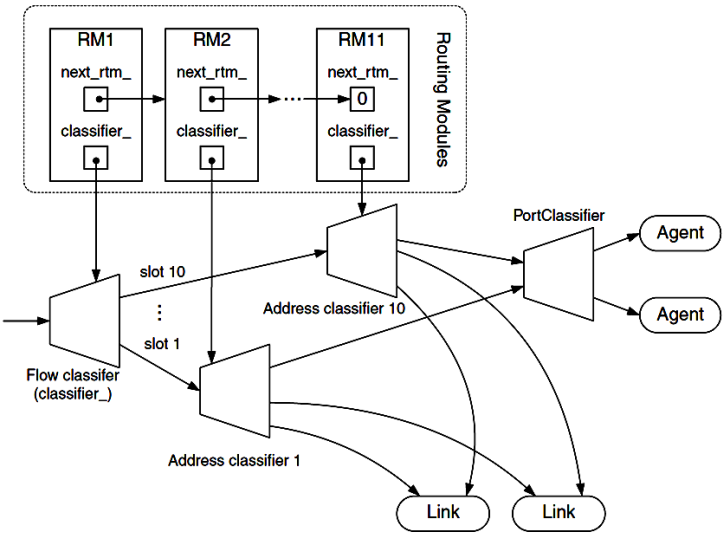


FIGURE 3.10: ns-2 routing module [122]

SoR because those two simulators were free, and have several aspects superior to those in other network simulators; in particular, both those simulators have fewer bugs. Consequently, the dissertation initially implemented the software SoR using ns-2. However, the ns-2 was deprecated by its successor, ns-3, and hence, the dissertation proposes to improve the software SoR using ns-3.

3.2.1 Main components required in software SoR

One of the primary motivations in developing the software SoR is to provide the functions of a general router such as packet forwarding, supporting TCP/IP protocol stack, and maintaining a routing module, which supports general routing protocols. Moreover, the software SoR should be capable of analyzing the packet payloads up to the application layer information, classify the packet streams, and store the necessary information in databases as shown in Fig.3.2. Furthermore, the databases should be accessible to the application module to enhance the user services. Consequently, as presented in Fig.3.9 the Software SoR comprises of:

1. A routing module including routing table and packet forwarders
2. A packet analyzer module
3. A recommendation module to provide content-based services
4. A database to store necessary packet stream information

Following sections elaborate each module in detail.

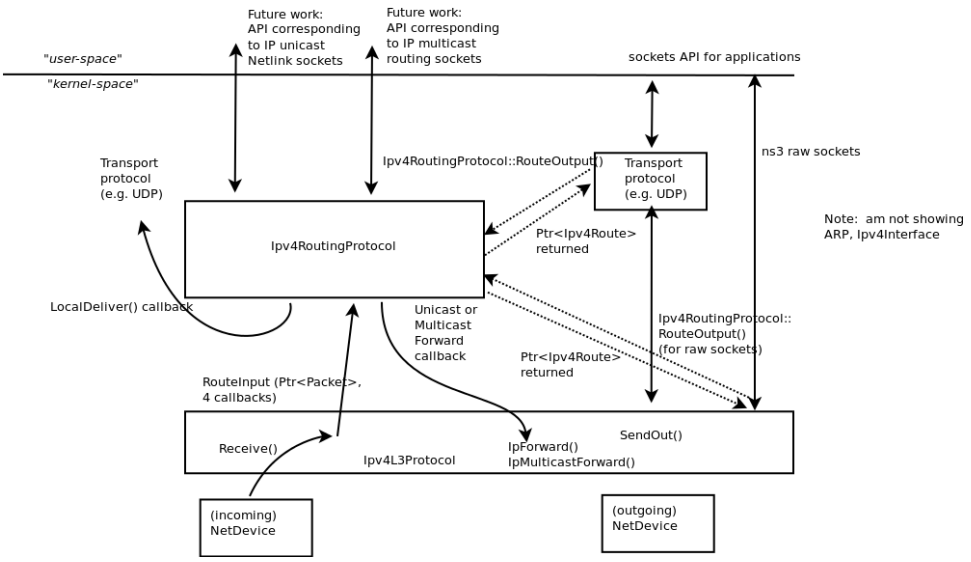


FIGURE 3.11: ns-3 routing module [123]

3.2.1.1 Routing module

The routing module is responsible for finding the network routes (e.g., shortest path, minimal delay path, or low-weight path) and forwarding packets to the next hop towards their destinations. Mainly, as depicted in Fig.3.9, the routing module is the combination of network interfaces, packet receivers, forwarders, and routing tables (Forwarding Information Bases). The software SoR is implemented by assuming the routing module available in the simulators, ns-2 and ns-3.

Figures 3.10 depicts the ns-2 routing module. As the figure elaborates, the main functionality of the ns-2 routing module is to facilitate the address classifiers [122]. The address classifiers are connected by maintaining a linear topology. Even if the topology of the classifiers is as complicated as a full mesh topology, the routing modules always remain linear and straightforward. Ns-2 facilitates configuration commands to the first routing module in line and let the routing modules propagate the configuration commands towards the end of the line. As stated in [122], and depicted in Fig.3.10, since all address classifiers are connected to one of the routing modules, the configuration command will eventually reach all classifier. Conceptually, ns-2 permits to create and associates a classifier with a routing module, and create a group of address classifier and a routing module. The classifiers can be configured through the "head" of the group (of routing modules) [2].

Figure 3.11 illustrates the ns-3 routing module. According to [122], unless the ns-2, the ns-3 module is designed to support traditional routing approaches and protocols, support ports of open source routing implementations, and facilitate research into unorthodox routing techniques. Moreover, [122] claims that the overall architecture presented in Fig.3.11 is designed to support different routing approaches, including (in the future) a Linux-like policy-based routing implementation, proactive and on-demand

TABLE 3.4: Routing protocols tested using proposed SoR module

ns-2		ns-3	
Protocol	Status	Protocol	Status
General	Yes	OSPF	Yes
AODV	No (ad-hoc)	Static	Yes
DSR	No (ad-hoc)	Global routing	Yes
DSDV	No (ad-hoc)	DVRP [97]	Yes
Static	yes	SLR [96]	Yes
		ESLR [96]	Yes
		AODV	No (ad-hoc)
		DSR	No (ad-hoc)
		DSDV	No (ad-hoc)

routing protocols, and simple routing protocols for when the simulation user does not care about routing. The basics of the ns-3 routing module class "Ipv4RoutingProtocol ()" declares a minimal interface, consisting of two methods:

1. RouteOutput
2. RouteInput

The transport protocol queries Ipv4 for the Ipv4RoutingProtocol object interface, and will request a route via RouteOutput function for packets traveling outbound from a host. This process returns a pointer to Ipv4Route object, and this is equivalent to a "dst_cache" entry in Linux routing module. The SoR is tested for the compatibility of the list of the routing protocols available in both simulators. Table 3.4 provides the available routing module and the capability of SoRs to support them.

3.2.1.2 Packet analyzer module

The packet analyzer module represents the "information extraction and collection" module of the general SoR presented in Fig.3.2. The analyzer module is implemented to analyze the packets up to

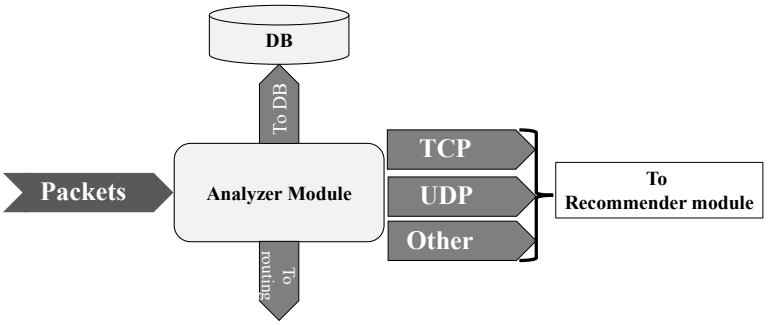


FIGURE 3.12: Functional diagram of analyzer module

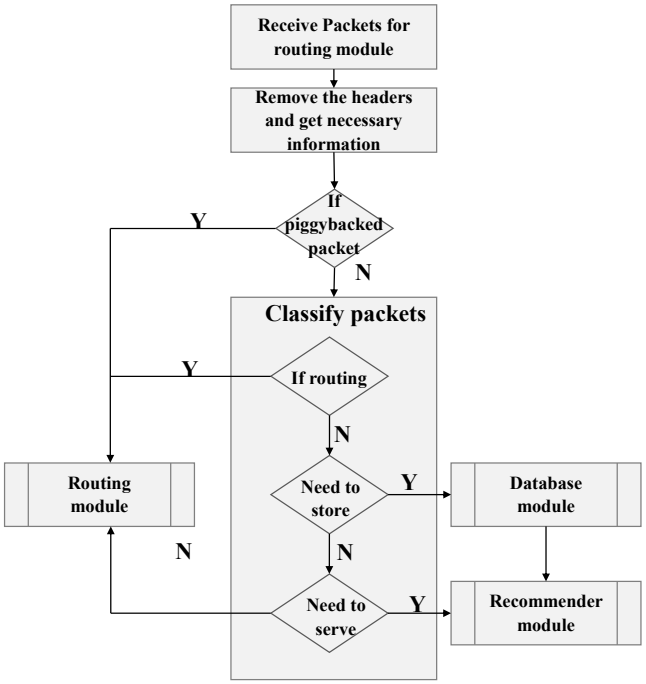


FIGURE 3.13: Instance of the analyzer module

Transport later and extract the necessary information. Figure 3.12 depicts the functional diagram of this module. As stated previously, the main function of this module to classifies the packets according to the five tuples: Source IP, Source Port, Destination IP, Destination Port, and the protocol by analyzing the IP header information. This module is designed and developed with the full capability of customization according to the requirement of the user, application, or the service. Moreover, this module is capable of application layer packet classification using the transport layer header information, e.g., distinguish DNS packets and read the header fields.

Figure 3.13 displays a general algorithm, which can be implemented into the analyzer module. Note that the analyzer module is developed with the scalability of changing the packet analysis algorithm to improve the programmability to support various applications. The reason is that, as pointed out in Section 3.1.4, the SoR is a general router that is capable of providing numerous services according to the requirement of the user, e.g., CDN, recommendation, IPS/IDS service. According to the Fig.3.13, firstly, the analyzer module receives a packet from the routing module. Analyzer module then removes the headers attached to the packet. After that, it classifies the packet streams according to the requirement of the user, application, or the service. Finally, it treats the packets as necessary; ignore the packets, store in databases, or send to the recommender module to further process the packets.

3.2.1.3 Recommender module

Recommender module represents the “Application” module of the general SoR presented in Fig.3.2. The recommender module is also implemented as a general module that permits alterations according

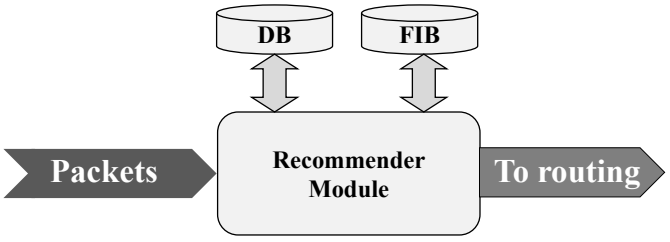


FIGURE 3.14: An instance of the recommender module

to the application, service, or the user requirement. In general, this module is capable of analyzing the packets, which are passed to the analyzer module, for the application layer information; analyze URL or URI of the data requests packets. Moreover, provide user services according to the application that the SoR is used for. Given that the primary motivation of this work is to enhance the user navigation process of CDNs, new functions such as analyze the DNS header information and create DNS recommendation messages are implemented into the recommender module. Moreover, since the SoRs are capable of forwarding packets based on the application layer information, the recommender module is implemented to changed the intended packet destinations by altering the destination IP address of IPv4 header, achieving on-the-fly packet redirection.

The high-level functional diagram of the recommender module is given in Fig.3.14. As depicted in the figure, the recommender module accepts the packets from the analyzer module and serves those packets accordingly. Moreover, this module communicates the available databases, which contains the necessary information for providing services, and the routing tables or the forwarding information bases to provide required services. The recommender module forwards the packets to the routing module directly when the requested task is completed. Note that a detailed explanation, including algorithms, of DNS recommendation, can be found in Chapter 5 Subsection 5.3 and IP address recommendation can be found in 5 Subsection 5.4.

3.2.1.4 Database module

Give that a database module is necessary to store the required information in the SoR, an in-memory No-SQL database module is proposed to the software SoR. As both prototypes of the SoRs were developed using C++ language, the databases were implemented using C++ map and multimap structures as key-value stores. Furthermore, the routing tables were also created by using the map structures. The keys were assumed according to the service the SoR programmed to provide. For an instance, the Key can be a hash value created according to the five tuples of each packet or the key can be the server ID or the CDN site ID.

3.2.2 Design software SoR using ns-2

The Main function required by the SoR is to monitor packet streams and perform DPI to the packets including payload and headers. Also, an SoR needed a unique routing function for on-the-fly packet routing according to the application layer information. However, ns-2 nodes are unable to support such advanced features. Therefore, as presented in Fig.3.15, ns-2 urged several modifications to its router core to support the data-rich functions of SoR. The primary ns-2 router is depicted in Fig.3.10. One main challenge towards implementing the SoR module on ns-2 was the ns-2 does not support packets structures with user data. Thus, several modifications were required to enhance the existing packet structures, packet/traffic generators, and transport layer agent to implement the SoR in the ns-2. The updated modules are depicted in Fig.3.16.

As shown in Fig.3.16, specifically, the ns-2 traffic generators were upgraded to support user-defined data from simulation scripts. Moreover, the traffic generator modules were further enhanced to generate traffic/packets according to a traffic generation rate given by users. The transport layer and network layer agents were also required several modifications to packetize the data received from the application layer traffic generator. The packetizing agents are also upgraded to support some functions such as trace packet characteristics, i.e., send time, receive time, RTT and packet size. Moreover, given that the existing ns-2 routers should be able to perform DPI and passive data collection to achieve SoR functions and dynamic/content-centric packet redirection, ns-2 router module was upgraded as depicted in Fig.3.17. That is, a new SoR packet analyzer module was implemented in the ns-2 node. Note that the ns-2 packet analyzer module assumes the works of both packet analyzer and the recommender module discussed in Section 3.2 (see Fig.3.9).

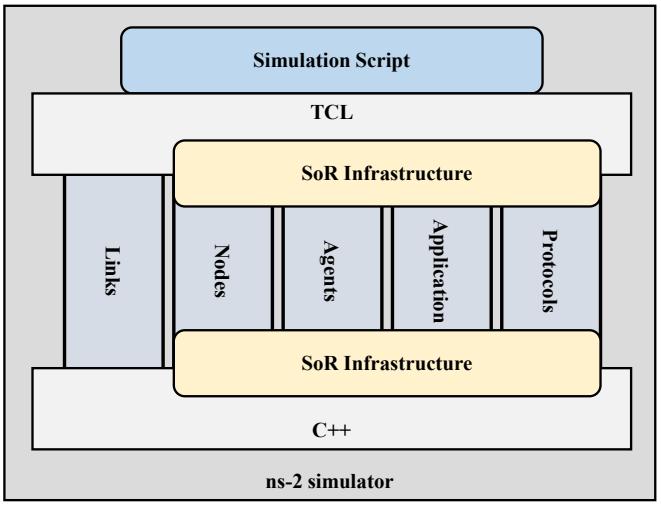


FIGURE 3.15: Modifications made on ns-2 simulator to support SoR architecture

3.2.2.1 ns-2 SoR architecture

As depicted in Fig.3.17, packet classifiers analyze IP headers to determine the next forwarding module to route the packets towards their destinations [122]. If packets are destined for the current node, port classifiers analyze transport layer headers and determine the port that the packets should be forwarded towards its corresponding application [122]. Therefore, the main challenge was to design the SoR packet analyzer such that it can perform DPI and content-centric packet rerouting as well as follow conditions above. Consequently, the SoR packet analyzer module was developed according to the properties of ns-2 virtual classifiers [122]. The SoR analyzer module was attached to the entry point,

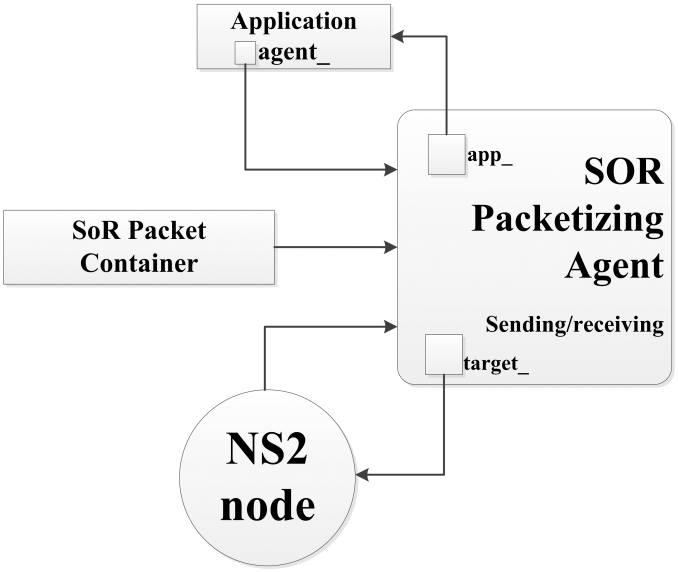


FIGURE 3.16: ns-2 SoR extension overview

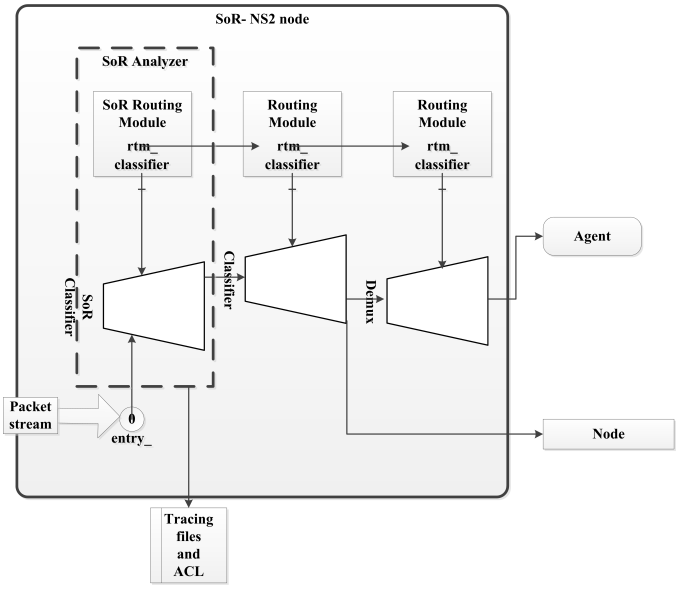


FIGURE 3.17: SoR Classifier and routing module attached to a ns-2 node

"entry_" of the ns-2 node (See Fig.3.17), and hence, always all packets have to go through the SoR. Ultimately, this feature made the ns-2 nodes work as SoRs.

When an SoR packet analyzer receives a packet stream, it analyzes the payload of the packet stream, modifies the packets if necessary, and passes the packet stream to the next classifier, which performs the actual routing. Consequently, the SoR packet analyzer can perform DPI for each packet that comes to the SoR before execute packet routing function. This is the main advantage of placing the SoR module next to the entry point of the ns-2 node. Moreover, this placement enables the SoR analyzer to change the packet header information without interrupting the routing. However, as shown in Fig.3.10, all classifiers required their routing modules to route packets accordingly. However, the SoR module is implemented to inherit the functions and methods of the immediate next classifier's routing module. This feature enabled the SoR to analyze the packet information and do the necessary alterations without depending on the routing module. Consequently, the routing module just forwards the modified packets to the next classifier. Both the classifier and routing modules were implemented using C++ by editing the core of the ns-2. Finally, as pointed out in Fig.3.15, those new modifications were mapped with the necessary OTcl modules to interconnect the ns-2 front-end with the ns-2 core.

In the early ages, network simulators do not necessarily simulate actual packet payloads due to memory management constraints. However, since the SoR is implemented to shift the current Internet infrastructure into service-based open innovation platform, the simulators required simulating actual packet details instead of the size and a pointer. Consequently, new ns-2 modules and classes were defined to emulate packets with payloads. Those defined classes were capable of manipulating a real data as specified by the user. In contrast, as depicted in Fig.3.16, several new modules were added to ns-2, namely, SoRPacketizingAgent, SoR traffic generators, and packet containers. Furthermore, the application modules were also modified to accept user data through the simulation scripts.

The packetizing agent, which is illustrated in the Fig.3.16, generates IP packets according to the traffic generation rate of the application layer traffic generator. This agent generates packets using the newly introduced packet structure, SoR packet container, which contains actual user data. The SoR agent gets the user-defined real payload from the application layer agent and adds it to the SoR packet container. SoR packetizing agent module then wraps the SoR packet with an IP header and sends it to the data link layer using the "sendmsg()" function, which is a virtual function inherited from the ns-2 agent class to capture the application data. Finally, SoR agent module sends the packet to the physical layer node through the "target_ pointer" [122, 123].

Ns-2 is deprecated by its successor, ns-3, and eventually, the SoR module was also required several modifications such as support generic header structures, Linux networking ready functions to practically evaluate the SoR applications in simulators, IPv4, and IpV6 support, and support real-world network topologies such as Rocket Fuel topologies. Moreover the developers also pointed out that the ns-2 SoR has lots of inherited ns-2 limitations. Among them, the most crucial limitation is that the ns-2 nodes have only one entry point to the node [122], and this limits the performance of the SoR module

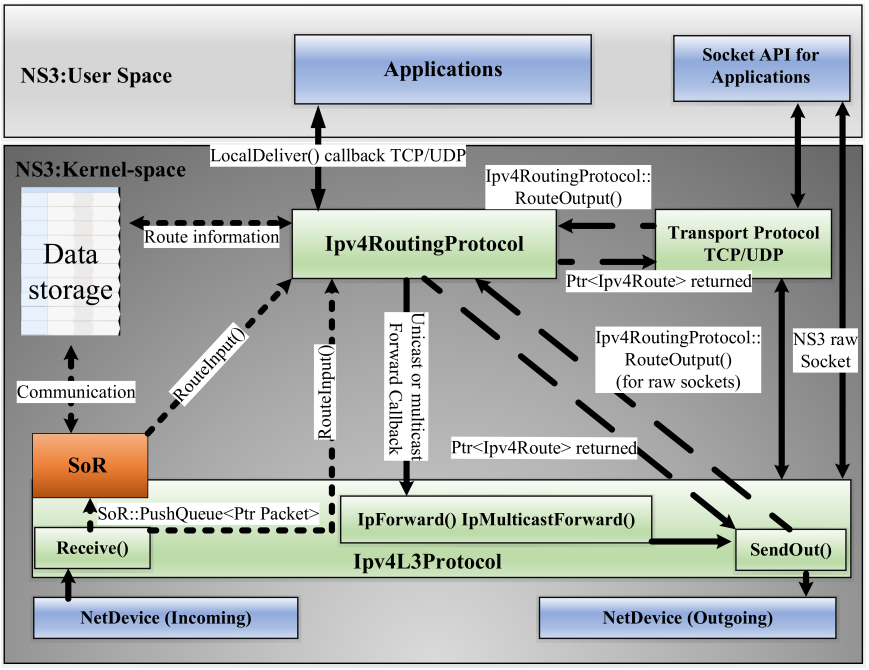


FIGURE 3.18: ns-3 node with the SoR extension [124]

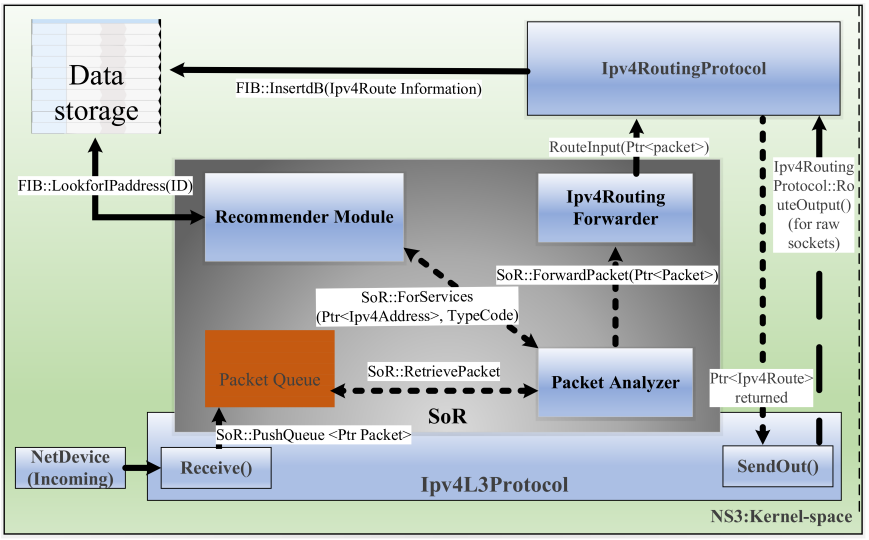


FIGURE 3.19: ns-3 SoR architecture

because the SoR could not maintain proper tracking of its network interfaces for the data analysis and redirection. This is cetainly limited the ns-2-SoR to support real-world applications after the DPI had been performed on the packets. Furthermore, as ns-2 supports only the ns-2 raw packet structure, it does not support payload generic real-world like packet structures. Consequently, the ns-2 SoR module could not function with real-world packet formats.

The network simulator 3 (ns-3) [121] is a discrete-event simulator that is proposed to replace the ns-2 simulator. Nodes in the ns-3 simulator are connected to each other using NetDevices communicating over their respective channels. A node can have multiple NetDevices as its interfaces. Nodes can

operate with or without an IP stack such as a Layer 2 switch or an Ethernet bridge. Furthermore, the packets in ns-3 can be serialized or desterilized to/from actual packet formats as they traverse through the network stack. This makes ns-3 well suited for real-world integration. Moreover, ns-3 has IPv4/IPv6 addressing schemes available for network specifications, something which ns-2 lacked and thus performed implicitly. These newly added features support our motivation to implement the SoR in a simulated real-world environment, making of ns-3, is well suited for the SoR implementation.

3.2.3 Design software SoR using ns-3

3.2.3.1 ns-3 SoR architecture

The implemented ns-3-SoR supports four basic features of the SoR [34, 35]

1. Capture packet at its interfaces.
2. Perform DPI to analyze the packet information including application layer data.
3. Detect shortest paths according to some pre-defined rule set and change the header information.
4. On-the-Fly packet redirection based on pre-defined set of rules.

As previously explained in the ns-2 SoR module, it is mandatory that the packets should initially pass the SoR module so that the SoR module can take the application decisions before a node takes the routing decisions. Therefore, as shown in Fig.3.18 SoR module was implemented to capture and buffer every packet coming to the incoming interface. This facilitates the proposed SoR to capture raw packets before they go through the protocol stack, and SoRs to have total control of the packet streams. Moreover, the SoR module supports another function, which is only to analyze the information by duplicating the packets and discarding them after finishing the analysis process.

$< nodename > -> EnableSoR(1)$; command was used to enable the ns3::Node for the SoR. If the SoRs are only configured to analyze the packets, once the packet arrives at the ns3::Node from the receiving ns3::NetDevice, *Node::ReceiveFromDevice()* function first duplicate the packet and passes to the packet into the SoR::packet queue. Meanwhile *Node::ReceiveFromDevice()* passes the packet to the routing module for perform L3 routing . When a packet arrives at the SoR packet queue, the SoR module retrieves the packet from the queue and analyzes the packet for the five tuples: sender IP, sender port, destination IP, destination port, and the protocol of the packet. Since the ns-3 is a modularized simulator and each module represent the layers of the TCP/IP stack, several modifications were done to the ns-3 datalink, The Internet, and Transport layers to implement the aforementioned method. The main challenge is to access the IP and transport layer header from the Datalink and Physical layers. This was because the ns3::networkModule does not have access to the ns3::InternetModule headers

[123]. Therefore, several changes were made to the Waf compiler to get access to the ns3::Internet module including the ns3::Ipv4header in the ns3::network module in order to perform the DPI in the SoR.

As shown in the Fig.3.19, the ns-3-SoR consists of a packet buffer, packet analyzer, shortest path detector module, databases, and Ipv4RoutingForwarder. Once a packet arrives at the ns3::Receiver, the receiver function calls the SoR::PacketQueue to store the packet in the SoR buffer. Then, the SoR::PacketAnalyzer module takes the packet and performs the DPI for the five-tuples. Finally, it displays the packet content in the standard display. Furthermore, the SoR packet analyzer module has the capability of altering the packet header information if necessary. Once a packet has been analyzed, if the packet needs a change of header, the SoR analyzer module does it according to the user requirement. For that SoR used the Shortest path detector module to get the best destination according to the configured routing and navigation policies. When the packet has been analyzed and altered, the packet is then passed to the ns3::Ipv4Routing Protocol for further routing. The ns3:: Ipv4Routing Protocol handles the packet according to the basic ns-3 packet forwarding processes.

3.2.4 Evaluate the proposed software SoR

3.2.4.1 In ns-2

Figure 3.20 shows the simple topology used for to simulate the modules created in ns-2, namely, Packet analyzer, packet generator, packetizing agent, and the header structures used to emulate payload. A constant rate packet generator was attached to an SoR agent and passed the payload data, “test_data” and “1234”, from the simulation script. Tracing outputs such as the number of generated packets, number of received packets, number of dropped packets, delay of packet delivery, and output of the SoR analyzer maintained accordingly to ensure end-to-end packet delivery of an SoR simulation topology with aforementioned traffic rate generation. The topology was configured as follow. The SoR packet generation agents were attached to Node 0. The NS2 typical packet generation agent was attached

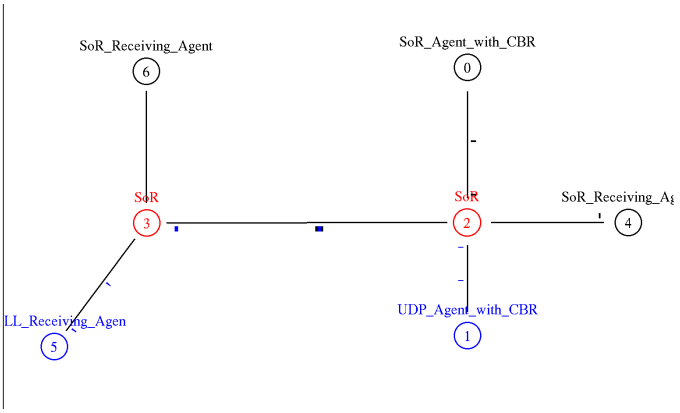


FIGURE 3.20: The topology used to validate SoR in ns-2

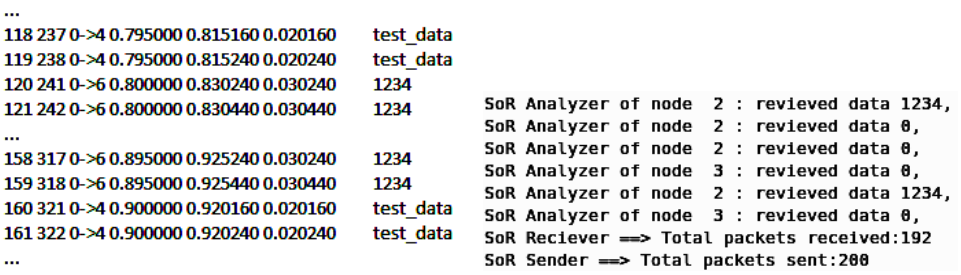


FIGURE 3.21: Sample output of SoR sender and receiver modules

to Node 1 using the CBR traffic generator. The SoR receiving agent was placed in Node 4 and the NULL sink receiving agent was placed in Node 5. Nodes 2 and three were configured as SoR nodes. In addition, an additional node, Node 6, was used to receive the dynamically rerouted packets from the SoR at Node 2. For the simplicity in traffic generation and protocol management, we assumed UDP protocol as the transport layer protocol and IPv4 as the IP layer protocol for the evaluation. The Same topology was created without using the SoRs for the comparison purposes.

Table 3.5 shows the comparison between the number of packets sent and received in the simulation with and without SoR. As shown in the table, the SoR sender and receiver display the number of packets generated and received, respectively. This verifies the successful end-to-end packet delivery and proper working of all modules. The results confirm that the packet drop rate is almost similar with and without SoR during the simulation time, 3000s. Furthermore, packet transfer delay of the packet streams was analyzed. The average data transfer delay was increased by 0.02 to 0.04ms in the SoR infrastructure. That was expected because the SoR analyze the packets for its payload and that consumes time. Overall results showed that the ns-2 SoR module was working without passing exceptions and without incurring additional memory or processor burdens.

Figure 3.21 shows that on-the-fly content-centric packet reroute has been achieved by the ns-2 SoR module successfully. The test case was implemented to analyze the packet payload and redirect the packets respectively. Initially, packets were created with a payload of "test_data" and packets were

TABLE 3.5: Evaluate the accuracy of ns-2 SoR module

	With SoR		Without SoR	
Time(sec)	Time(sec)	SoR sender	SoR Reciever	UDP sender
1.5	1.5	404	394	401
10	10	3802	3794	3800
20	20	7802	7794	7800
30	30	11804	11794	11802
40	40	15802	15794	15800
50	50	19802	19794	19800
60	60	23802	23794	23800
300	300	119804	119794	119802
3000	3000	1199802	1199794	1199800

destined to Node 4. The traffic generator changed the packet payload from "test_data" to "1234" and vice versa randomly. However, the destination address was kept unchanged. A new routing policy was added to the SoR 2 stating that if the packet payload is "1234," then route the packet to Node 6. Therefore, when the SoR 2 receives a packet with payload "1234", the destination address was changed accordingly. As displayed in Fig.3.21 starting from packet ID number 120 at 0.8th second, the traffic generator generated packets with content "1234." According to the figure, the packets shows that the destination address has been changed to Node 6. Moreover, the delivery time has been increased to 0.010080 seconds. After few seconds later, the traffic generator again changes the packet payload to "test_data," and as shown in the Fig.3.21 the packets destination was unchanged and the packets were routed to the intended destination, Node 4. This proves that the ns-2 SoR module is capable of redirecting packet based on the content of the packet, incurring that content centric network can be achieved using ns-2 SoR [36].

3.2.4.2 In ns-3

The simulation topology shown in the Fig.3.22 was implemented on a computer with an Intel i7 3.4 GHz processor with 8 GB of memory. All links were configured as 5 Mbps P2P with 2 ms link delay. The six NS3::nodes were capable of functioning as both SoR and regular NS3::Nodes and the data traffic was handled using a UDP echo client and servers. First, every UDP echo client was programmed to send 5,000 to 50,000 packets to the UDP echo server and measured the average processing delay of both the SoR and ns3::Node. The measured results are displayed in Fig.3.23. According to the figure, on average, the SoRs consumed 4micro seconds for the packet processing, i.e., analyzing, routing, and forwarding, while the ns3::Node consumed only 1micro second. The SoR processing time was 3micro seconds higher compared to the NS3::Node because the SoR performed DPI to the packets to determine the 5-tuples. However, it is noticeable that the processing time maintained constantly, and the SoR is buffering module and the packet analyzer module work constantly without incurring any further delays. The SoRs were configured to output the packet analysis results into the terminal

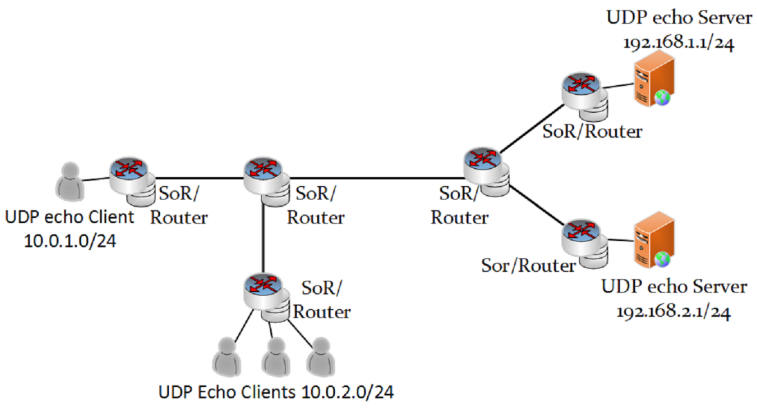


FIGURE 3.22: The topology used to validate SoR in ns-3

windows. The output is given in Fig.3.24. The figure clearly shows that the SoRs are capable of analyzing the packet data for the five tuples. Moreover, we reinforced a routing policy to the edge SoR of Client "10.0.1.1," SoR 1, to redirect the packet to IP address "192.168.16.1" if the packet is requesting for content ID "0001." Consequently, the URL would be rtsp://5.6.7.8:9/0001. As shown in the Fig.3.24 the SoR 1 stated hat it received a packet destined to "5.6.7.8." Since the requested content ID is "0001," the SoR1 changed the destination IP address to "192.168.16.1" and the rest of the SoR on the route path says that the destination is "192.168.16.1." This means that the SoR implemented in ns-3 is also capable of on-the-fly dynamic packet redirection. Moreover, the SoR can be accessed the FIBs according to the configured user redirection policies. Furthermore, The SoRs were tested for their memory usage and the simulation time usage. This experiment was conducted by cumulatively increasing the number of SoRs in the simulation topology. The obtained results are displayed in Table 3.6. According to the table, the SoRs consumed averagely 0.41% of the processor compared to the ns3::Nodes. Moreover, when SoRs were used in the simulation, the simulation time was increased on average 7s. These yielded results confirm that the proposed software SoR functions correctly without additional process and memory consumption, and the software SoR can be used to simulate the content necessary applications that the regular SoR is proposed to achieve.

Given that the ns-3 is capable of emulating networks with hundreds of nodes such as ISP topologies, the SoR were configured on such networks and measured the process and memory utilization using the same computer mentioned above. For this experiment, two ISP topologies, ASN1221, and ASN3967 were used. A detailed explanation about the ISP topologies can be found by referring Table 4.4. Moreover, hundred random packet generators were used and placed them according to the PoP information

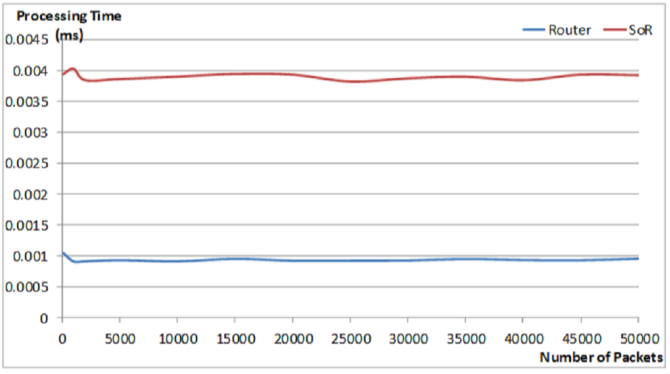
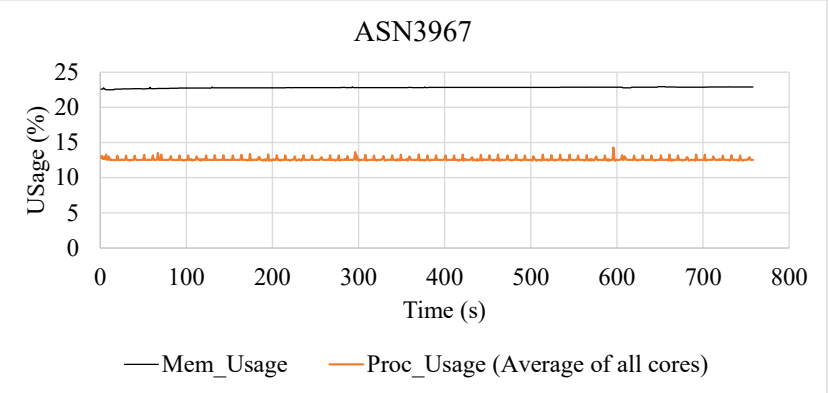


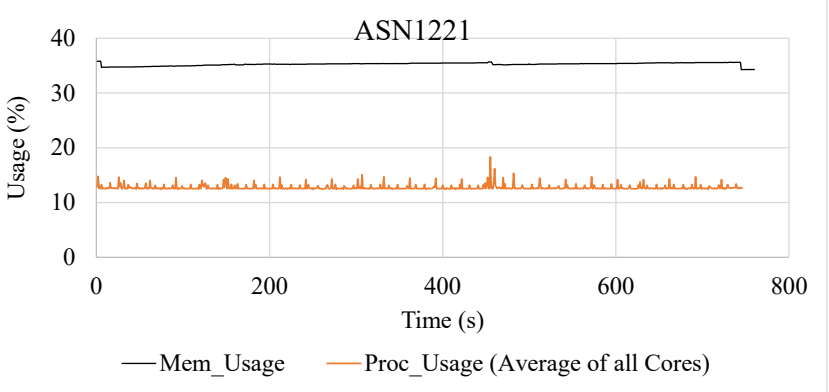
FIGURE 3.23: Processing time comparison with a regular ns-3 router

```
1: At time 2s client sent 20 bytes to 5.6.7.8 port 9
This is from the Node:1 UDP packet 10.0.1.1:49153 --> 5.6.7.8:9
This is from the Node:2 UDP packet 10.0.1.1:49153 --> 192.168.16.1:9
This is from the Node:3 UDP packet 10.0.1.1:49153 --> 192.168.16.1:9
At time 2.0063s server received 20 bytes from 10.0.1.1 port 49153 and
At time 2.0063s server sent 32 bytes to 10.0.1.1 port 49153
This is from the Node:2 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
This is from the Node:1 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
This is from the Node:0 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
At time 2.0126s client received 20 bytes from 10.0.3.1 port 9 and the
```

FIGURE 3.24: Five tuple analysis capability of SoR in ns-3



(I) Processor and Memory usage of ASN3967



(II) Processor and Memory usage of ASN1221

FIGURE 3.25: Processor and memory usage for ns-3 SoR in large topologies

TABLE 3.6: Processor usage and the total simulation time respect to number of SoRs

SoR/Node amount	Processor Utilization (%)	Simulation Time (s)
6 Nodes	97.79	59
1 SoR	97.95	58
2 SoRs	99.05	60
3 SoRs	98.02	65
4 SoRs	98.69	70
5 SoRs	97.59	72
6 SoRs	98.1	70

of those ISP. The traffic generators were configured to send burst traffic conditions to make the SoRs busy. Furthermore, ESLR routing protocol is assumed in those topologies. Consequently, process usage and memory usage were measured for 1,000 s simulations. Obtained test results are presented in Figs.3.25. The yielded results show that even with more than 200 nodes, the proposed SoR module is stabilized, and the memory and process usage is optimized. It is notable that the memory usage of ASN1221 is higher compared to the ASN3967. The reason is that the number of routers in ASN1221 is high, and a number of links are also comparatively high. Therefore, the memory consumption is high for SoR packet analyzing is comparatively high. The latest version of the implemented ns-3 SoR is available to download from this reference [38, 104].

Chapter 4

Server Link Router-state Routing Protocol

Network path selection is the process that maximizes network resource utilization and minimizes network congestion for efficient packet routing in ISP networks [125, 126]. Moreover, as explained in Chapter 1, network path selection is one attribute of the content navigation. Hense, effective content navigation is highly contingent of the effective network pth selection. ISPs use interior gateway protocols (IGPs) for network path selection according to the prevailing network link conditions. Current IGPs, i.e., "shortest path routing protocols," are optimized for a single arbitrary metric, administrative weight [75, 125, 126]. Furthermore, as discussed in Chapter 2, CPs deploy their networks within the ISPs networks, e.g., telcos [16]. In such networks, CPs maintain distributed server placement structures and redirect their subscribers among the content servers according to either proximity or end-to-end latency [18, 74, 127]. Therefore, primary assumptions of network path selection, i.e., the traffic matrix is point-to-point and constant, are violated [27, 73].

Moreover, as stated in [73], when an ISP optimizes its IGP for a particular CP, the IGP confines to consider non-CP traffic for network path selection. However, as illustrated in Fig.1.4, both procedures involved in user navigation: service point selection and network path selection, are interconnected. To this end, it is important for IGPs to contemplate the dynamicity introduced by the CPs for network path selection. Behavioral patterns of network devices: routers and links, signify the dynamicity introduced by CPs for the ISP networks [73]. It is important that ISPs optimize their IGPs using network device states as parameters of network path selection rather than optimizing IGPs for a single arbitrary metric for network path selection. Nevertheless, given that, in modern networks, CPs are heavily contingent of ISPs, CPs should consider route matrix for server selection to ensure appropriate user navigation in CDNs. Subsequently, IGPs should interpret the route matrix in a way that CPs can use it, and thus, IGPs should provide a technique to access the route matrix for server selection.

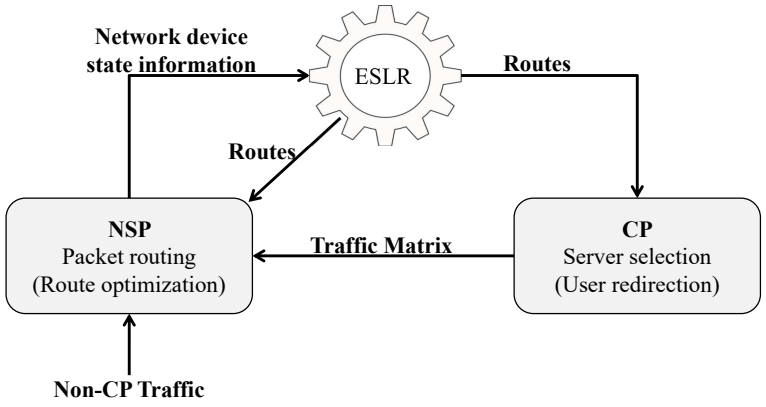


FIGURE 4.1: The basic concept of SLRouting protocol

TABLE 4.1: Assumption made to represent device state using "time"

Device	Connect state	Disconnect state
Links	0<`propagation delay`<∞	`propagation delay` = ∞
Routers	0<`processing delay`<∞	`processing delay` = ∞
Servers	0<`processing delay`<∞	`propagation delay` = ∞

To this end, Server Link Router-state Routing (SLRouting) Protocol forms a hypothesis to use network device state information for network path selection [96]. The SLRouting protocol is then updated to the Extended SLRouting protocol (ESLR). The basic system diagram is presented in Fig.4.1. As illustrated in the figure, the SLRouting protocol is developed to provide three key services:

1. Use network device states as a parameter for route metric calculation.
2. Use dynamicity of the traffic matrix as a parameter for route metric calculation.
3. Provide an interface for CPs to use the route metric for server selection.

The SLRouting protocol proposes a method for converting network device states into a standard scale, time. Thereby, the SLRouting protocol selects an effective path (ep), which is the minimal delay path between two destinations, as the best route.

As illustrated in Fig.4.1, the proposed SLRouting protocol defines the network device state in "time" scale because the impact of network devices: routers, links, and servers, for network path selection can be characterized using "delay" as the common factor [75, 93, 96]. For example, influence of a router can be inferred as the delay introduced by the routers for processing and forwarding a given packet. The impact of network links can be considered by measuring link loads and link usage information [128]. The impact of the CP traffic matrix for network path selection can be interpreted as the burden created by the traffic matrix for both network links and routers [27, 73]. Moreover, in peer-to-peer networks, as explained in [93], the servers are also an important parameter for network path selection. Hence, SLRouting facilitates using a server as a parameter for network path calculation. Consequently,

network states can be interpreted by using "time" as the parameter. For an example, as presented in Table 4.1, if the links are considered as in the connected state, the link propagation delay should satisfy 0<`propagation delay`<∞, and if the link is in disconnected state propagation delay is considered as infinity. Similarly, both routers and servers can be defined using the same method explained in Table 4.1.

Spring et al. indicated that intra-domain network path selection is latency-sensitive [129], suggesting that solving delay-based routing is a difficult, but significant, achievement for many future networks [130]. Subsequently, control techniques such as diffusion computation [96], coordinated messaging [129, 131], push and pull routes, and hold-down timers were used to implement in to the extended version extended SLRouting Protocol (ESLR) for stable delay-based routing. The main challenge in carrying out a delay-based routing protocol is to maintain a method that can compensate the route oscillation problem. As explained in [92, 131], the route oscillation problem is the result of flooding link state information and recalculating the topology table at every instance of topology change. For example, if the link between $k1$ and $k2$ frequently marks its state as ON and OFF (see Fig.4.2), the rest of the nodes should recalculate the topology table every time they receive route advertisement. Such scenarios probably consume a significant percentage of the bandwidth of network links. Certainly, the question is how to overcome such situations.

One method used in ESLR is periodic advertisements, instead of link-state advertisements [1]; route advertisements do not flood at each state change of the network device. The other method is for ESLR to use time delay, i.e., settling time, to maintain route records in a settling stage before such records are considered as valid and constant routes [131]. During the settling time, ESLR determines whether to ignore or manage newly received route records accurately without affecting the stable route topology information.

Given that distance vector algorithms use the aforementioned methods for effective route management, in ESLR, the Bellman-Ford algorithm is used for route matrix computation [132]; the main intention is to prevent the route oscillation problem. However, ESLR inherited the limitations of the Bellman-Ford algorithm, such as slow convergence and count to infinity. ESLR is designed by using the diffusion computation concept [96] and coordinated update approach [129, 131] in order to maintain effective and stable route computation and distribution by compensating for such limitations. A detailed explanation of the control techniques used in ESLR is provided in Section 4.3.

Consequently, this chapter elaborates the route calculation methods, algorithms, and implementation notes with an evaluation of the proposed protocol using two Rocketfuel [133] ISP topologies [133] Exodus-USA (AS3967) and Telstra-AUS (AS1221).

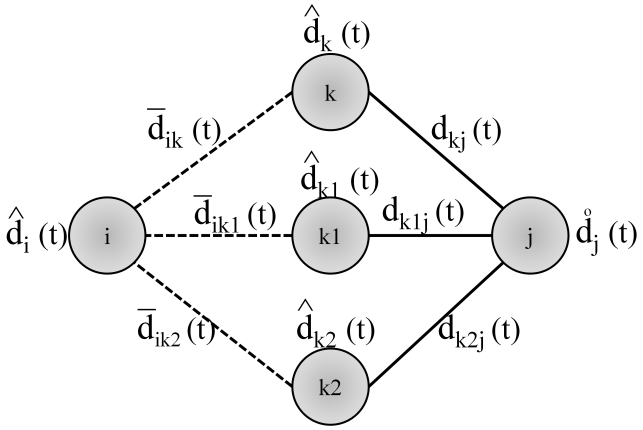


FIGURE 4.2: Symbolic representation of routers, links, and servers (solid lines denote direct links and dotted lines denote the distance)

TABLE 4.2: Notations used in equations

Notation	Explanation
j	Servers
k	Routers
i	Source nodes
$\hat{d}_k$	Average time a given packet has to wait at the router
d_{kj}	Average time a given packet travel through the link
$\mathring{d}_j$	Average time a given packet has to wait at the server
ep	Effective path (minimal delay path)

4.1 Using Bellman-Ford algorithm for route computation

Let us use the parameters given in Table 4.2 and symbolize representation of servers, links and routers presented in Fig.4.2 for the explaining the network devices states, i.e., time, calculation on ESLR. Consider a network represented by Graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of directed physical links. Let M_{ij} denote the cumulative traveling delay of flow (i,j) from node i to j, where $i,k,j \in V$. Flows are carried on end-to-end paths consisting of intermediate routers and links, and M_{ij} is the summation of the propagation delay through the intermediate routers and links.

According to the explanation given in [125, 134] a server can be denoted using the M/M/1 queue, where the Poisson process determines the arrival rate (λ_S), and job service (μ_S) has an exponential distribution. Using the detailed explanation given in [125, 126] and the Little's formula [135], at the steady state, the packet waiting time ($\dot{d}_j(t)$) of a given server j can be calculated using Eq.(4.1), where j_n denotes the server index and $n = 1, 2, 3, \ldots$

$$\dot{d}_{j_n}(t) = 1/(\mu_{j_n}(t) - \lambda_{j_n}(t)) \tag{4.1}$$

Reference [134] explains that routers are also developed on the basis of the queue model. Assuming

a Poisson packet arrival rate (λ_R) and exponentially distributed service rate (μ_R), similarly, routers can also be observers by using the M/M/1 queue model. The packet waiting time ($\hat{d}_k(t)$) of a given router k can be calculated using Eq.(4.2), where k_n denotes the router index and $n = 1, 2, 3, \ldots$

$$\hat{d}_{k_n}(t) = 1/(\mu_{k_n}(t) - \lambda_{k_n}(t)) \tag{4.2}$$

The packet traveling delay of a link depends on two major points:

1. packet transmission delay (T_{trans}) and
2. packet propagation delay (T_{prop})

T_{trans} of a link kj (see Fig.4.2) depends on the available bandwidth $L_b(t) = L_c - L_l(t)$ of a link, where L_l denotes the used bandwidth and L_c represents the total bandwidth [128]. In this case, for a packet of size Z propagates through the link, and T_{trans} can be calculated using Eq.(4.3).

$$T_{trans}(t) = Z/L_b(t) \tag{4.3}$$

Further, T_{prop} of a link depends on the length and the medium of the link [128]. Consequently, total traveling delay ($d_{kj}(t)$) of a given link kj can be written using Eq.(4.4).

$$d_{kj}(t) = T_{prop_{kj}}(t) + T_{trans_{kj}}(t) \tag{4.4}$$

Now, assuming the scenario presented in Fig.4.3, at the time t, the cumulative packet propagation delay $\mathbb{T}_{k_2}(t)$ between k_1 and j via router k_2 can be obtained using Eq.(4.5), where $\bar{d}_{k_2k_4}(t)$ represents the minimal traveling delay between distance nodes k_2 and k_4.

$$\mathbb{T}(t) = \left[\dot{d}_j(t)\right] + \left[d_{k_1k_2}(t) + \bar{d}_{k_2k_4}(t) + d_{k_4j}(t)\right] + \left[\hat{d}_{k_2}(t) + \hat{d}_{k_4}(t)\right] \tag{4.5}$$

Similarly, at the time t, the cumulative packet propagation delay $\mathbb{T}_{k_3}(t)$ between k_1 and j via router k_3 using Eq.(4.6), where $\bar{d}_{k_3k_4}(t)$ represents the minimal traveling delay between distance nodes k_2 and k_4.

$$\mathbb{T}(t) = \left[\dot{d}_j(t)\right] + \left[d_{k_1k_3}(t) + \bar{d}_{k_3k_4}(t) + d_{k_4j}(t)\right] + \left[\hat{d}_{k_3}(t) + \hat{d}_{k_4}(t)\right] \tag{4.6}$$

4.2 Determining ep using the distributed Bellman-Ford algorithm

Given that current ISP are heavily contingent of dynamic behavior [81, 136] introduced by CPs, and thus, IGPs should consider such behavior for network path selection, the router metric calculation has been done by considering the time domain in order to contemplate the time shifts of traffic matrix.

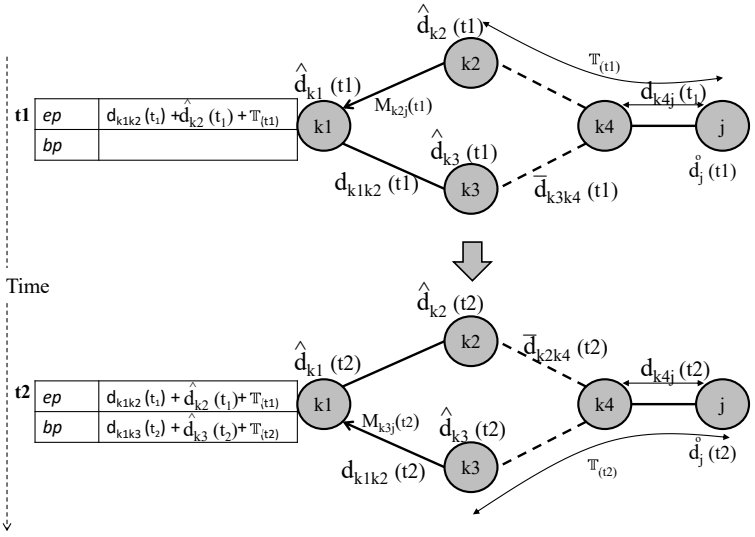


FIGURE 4.3: Using distributed Bellman-Ford algorithm to calculate *ep* (solid lines denote direct links and dotted lines denote the distance)

Consequently, ESLR uses the distributed Bellman-Ford algorithm to calculate *ep*s according to the network state information at a given time t. Theoretically, introduction of the time domain dependency for metric calculation eventually leads to route oscillations. As stated earlier, however, ESLR confines the route oscillation problem by calculating and advertising the metric at distinct time intervals. The scenario presented in Fig.4.3 illustrates the use of the distributed Bellman-Ford algorithm to calculate the minimal delay path (*ep*). In general, the distributed Bellman-Ford algorithm can be written using Eq.(4.7) for a given time t, where $\mathbb{T}(t)$ denotes the minimal delay between the two given destinations, i.e., i and j (see Fig.4.3) [125].

$$\mathbb{T}_{ij}^{k_1}(t) = min\{\bar{D}_{k_1 j}(t) + d_{ik_1}(t)\} \tag{4.7}$$

Now, as illustrated in Fig.4.3, let us suppose that node $k1$ receives two route updates at $t1$ and $t2$. At time $t1$, $k1$ receives a new destination, j, from $k2$. $k1$ then calculates the cumulative propagation delay using Eq.(4.8), where $\mathbb{T}_{t1} = \bar{D}_{k_2 j}^{k_4}(t1)$. $k1$ considers the calculated cumulative time as the best path to the new destination j because this is the first route for destination j.

$$M_{k_1 j}(t1) = d_{k_1 k_2}(t1) + \hat{d}_{k_2}(t1) + \mathbb{T}(t1) \tag{4.8}$$

Similarly, at time $t2$, when k_1 receives route advertisement about destination j from k_3, the new propagation time through k_3 is calculated using Eq.(4.9), where $\mathbb{T}(t2) = \bar{D}_{k_3 j}^{k_4}(t2)$.

$$M_{k_1 j}(t2) = d_{k_1 k_3}(t2) + \hat{d}_{k_3}(t2) + \mathbb{T}(t2) \tag{4.9}$$

Assuming that $M_{k_1 j}(t1) < M_{k_1 j}(t2)$, k_1 uses the route via k_2 as ep, and the route via k_3 as the backup path

for destination j. Subsequently, at a given time t, i.e., at a time anode receives a route advertisement, the composite metric for a destination j from a node i can be simplified as Eq.(4.10), where n and m are the indexes of the network components in ep, $n \neq m$, $i \neq j$, and i can be considered a router.

$$M^{k_n}_{ep_{(i,j)}}(t) = K1\Big[\dot{d}_j(t)\Big] + K2\Big[\sum_{n,m \in ep} d_{k_n k_m}(t)\Big] + K3\Big[\sum_{n \in ep} \hat{d}_{k_n}(t)\Big] \tag{4.10}$$

For more accurate and better route calculation, ESLR allows controlling the default parameters using three non-negative controllable coefficients values: $K1$, $K2$, and $K3$ (henceforth referred to as $CCVs$). The CCVs represent enable, disable, and a scale level in which the parameters are used in the composite metric calculation. Values 1 and 0 represent enable and disable of each parameter. In addition, the current implementation supports CCVs up to eight scale levels.

The $k1$ value is specifically introduced to allow servers to participate in network path selection. As stated previously, there are several applications where IGPs require servers as a parameter for network path selection. For example, in P2P networks (CDNs), overlay networks (VPN), and data center networks, long distance links appear to be a single hop for IGPs. In such cases, ESLR facilitates using server state information to determine the minimal delay path by setting $K1$=1. It must be noted that ESLR assumes it is the responsibility of administrators to configure a content or management server with the gateway address; consequently, the servers periodically advertise the state information to the corresponding gateway router. If $K2$ is enabled, ESLR uses the link propagation delay for composite metric calculation. As pointed out earlier, ESLR uses the link capacities and available theoretical bandwidth of the interfaces to measure link transmission and propagation time. Furthermore, ESLR considers the available link bandwidth as a one-way-based value computed using a factor of the links' capacity. As already mentioned, network middleboxes (i.e., routers) are becoming pervasive on the Internet for providing user services, such as recommendations [137], firewalls, and DPIs [60, 61]. Consequently, ESLR uses the states of the routers to calculate the composite route metric by introducing the $K3$ coefficient value. Thus, by controlling $K3$, network administrators can decide the influence scale of routers considered for cost calculation.

The default CCVs $K1$=1, $K2$=1, and $K3$=1 have been prudently selected to provide optimal performance in most networks. The adjustment of CCVs allows per-deployment (i.e., ISP or CP) tuning of the network components used for composite metric calculation.

4.3 Using diffusion computation concept and coordinated updates for network path selection

The diffusion computation concept is introduced in [138] to calculate loop-free network paths every time regardless of the operation conditions of ISP or the Internet. Coordination update (advertisement)

TABLE 4.3: Acronyms used in ESLR for explanation

Acronym	Name
N-table	Neighbor table
M-table	Main routing table
B-table	Backup routing table
m-route	Main route (ep)
r-route	Reference route for the m-route in (in B-Table)
b-route	Backup route for the m-routes (in B-Table)
RRQ	Route request message
RRS	Route response message
SRC	Server-Router communication message
Hello	Hello message
KAM	Keep alive message
VALID	Neighbor and route records in the Valid state
INVALID	Invalid route records
BROKEN	Route records about broken networks
DISCONNECTED	Route records about either unresponsive local interfaces or neighbors

activities have been proposed to ensure proper message flow among nodes within a given topology [1, 139]. With the aid of the diffusion computation and coordinated update approach, in order to ensure stable and accurate use of network device state information for network path selection, ESLR is designed based on three important factors:

1. Within a finite period, a node should be able to determine the existence of a new node or connectivity loss of a neighbor
2. The protocol should ensure the reliability of the route update messages
3. Any route message event should be processed individually.

ESLR maintains a neighbor management module to ensure the first factor, whereas it maintains a route management module to make use of the second and third factors for better network path selection. Consequently, the general methodology elaborated in this paper can be divided as follows: 1) neighbor management, 2) route management, 3) coordination activities in link breakdown situation, 4) oscillation compensation, and 5) protocol implementation. The acronyms used for the explanation are given in Table 4.3. The current ESLR implementation is intended to allow routers to exchange information for computing routes using IPv4-based networks. The protocol is implemented with the support of both multicast and unicast route update methods. Furthermore, ESLR is developed as a UDP-based protocol and uses UDP ports 275 and 276 for its route advertisements [96]. The header structures used in this implementation are also available in [96].

4.3.1 Neighbor management module

The diffusion computation concept explains that each node should check for the connectivity and the availability of its neighbors periodically. Therefore, in ESLR implementation, each node in the topology maintains a list of neighboring/adjacent nodes to ensure loop-free route management. The neighbor management module consists of:

1. neighbor discovery
2. neighbor record maintenance modules

Neighbor module uses two type of advertisement messages:

1. Hello message
2. Keep Alive Message (KAM)

Both messages use the header structure presented in Fig.4.4. The management module required each node to maintain a neighbor table (N-Table) using the structure given in Fig.4.5. The N-Table facilitates methods to add, update, validate, and delete neighbor records adequately.

ESLR initiates neighbor module when nodes (interfaces) are first activated. Assume the scenario presented in Fig.4.6 in the following explanation. When *i* initializes its interfaces, it broadcasts *hello* messages among its interfaces. When *k1* receives a *hello* message from *i*, it generates and sends a *hello* message to *i* by setting the same identification number and authentication details received from *i*. Then *k1* waits for *NbrTimeout* time until it receives an *RRQ* message from *i*. If *k1* does not receive *RRQ* messages, it discards the record of *i*. Otherwise, *k1* sends an *RRQ* message by requesting the entire

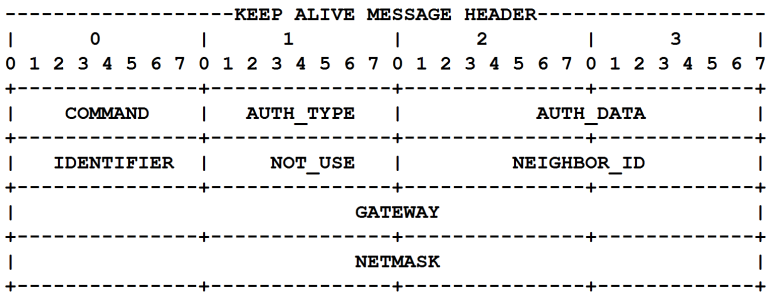


FIGURE 4.4: Hello/Keep Alive Message header

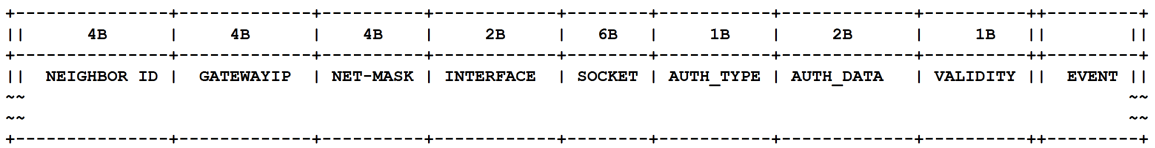


FIGURE 4.5: Neighbor table structure

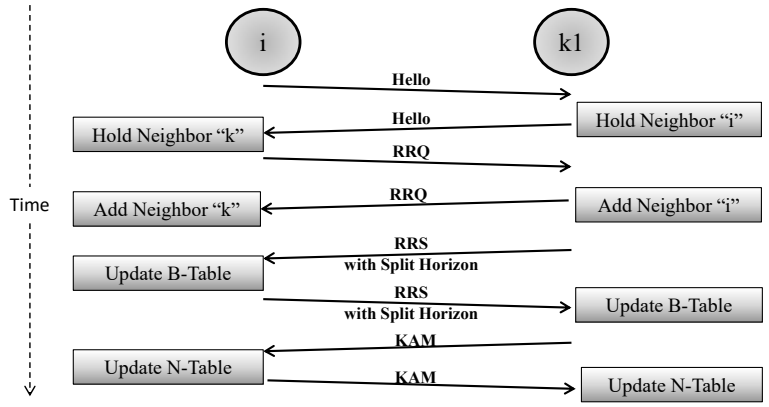


FIGURE 4.6: Neighbor discovery process

routing table of *i*. Finally, after both *i* and *k1* exchange their *M-Table*s, at each *KAMTimer* interval, both nodes exchange *KAM* messages to check their connectivity. Note that all timer values are given in Table 4.6

In ESLR, each node maintains a *N-table* to maintain a record of its neighbor nodes. As shown in Fig.4.5, each record contains information about the neighbors, including their IP addresses, network masks, authentication details, local socket information, local interface information, and validity. Each neighbor record is associated with two timer events, *NBRTimeout* and *Garbage-colct*, for effective neighbor maintenance (default values are listed in Table 4.6). All new neighbor records are set as *VALID*, and events are scheduled to expire after *NBRTimeout* seconds. The timeout event of that particular neighbor record will be re-set when a node receives a *KAM* from the corresponding neighbor node. However, ESLR marks neighbor records as *INVALID* and, schedule an event to remove that neighbor records from the *N-Table* after *Garbage-collection* seconds if the neighbor records do not receive *KAM*s until the timeout event expires. Moreover, ESLR invalidate all route records which refer to the invalidated neighbor as their gateway (a detailed explanation can be found in Section 4.3.2.1)

4.3.2 Route management module

The route management module consists of following functions:

1. Routing tables
2. Route advertisement headers
3. Route discovery
4. Route advertisement processing methods
5. Route invalidation methods.

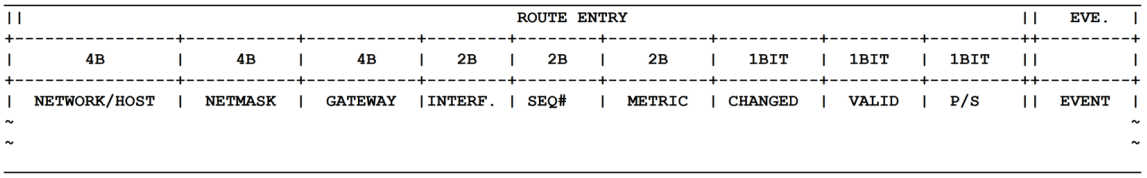


FIGURE 4.7: ESLR route table and attributes

ESLR uses techniques such as split horizon [140], poison reverse [140], periodic and triggered advertisements, and timer events, for faster, reliable, and stable route management. The default values for the timers are listed in Table 4.6. ESLR uses *push* and *pull* message techniques, i.e., coordination activities, for quick recovery [75]. *push* messages can be categorized as *hello*, *KAM*, and *RRS*, whereas *pull* messages can be categorized as *RRQ* messages. Moreover, ESLR distinguishes route advertisements using sequence numbers for consistent route propagation in large networks. When updates are flooding across the network, the nodes can check the sequence number to identify the most up-to-date route advertisements and discard the outdated advertisements.

4.3.2.1 Route tables and route records

ESLR maintains two routing tables, main (*M-table*) and backup tables (*B-table*), for consistent and stable route management, to address the route oscillations problem in delay-based routing, and to maintain fast loop-free route recovery. The table structure and the attributes are given in Fig.4.7. *M-table* maintains all *ep*s (*m-route*s) for each destination that can possibly be reached from the node. *B-table* maintains two route records for each destination listed in *M-table*. The first is *r-route*, a reference route to *m-route*. The second is *b-route*, which has the next best propagation delay compared to the up-to-date *r-route*, i.e., might higher than *m-route*.

The *b-route* is particularly used to prevent the route oscillation problem of delay-based routing approaches. When a link is disconnected, or a node is not responding, the routers use the *b-route* for quick recovery without sending coordinated messages among its neighbors. Furthermore, *b-route*s are used in this implementation to validate the consistency and the integrity of the *m-route*s. In any case that a node found that, i.e., route updating procedure, the *b-route* records for a particular destination is better than that of the *r-route*, the router quickly changes both *m-route* and *r-route* records in order to offer the minimal delay path. However, this procedure may cause heavy route oscillations and micro loops. Subsequently, ESLR maintains proper timeout values to update route records to prevent quick exchanges of topology table, i.e., *m-Table*; thereby, maintain consistency among the route records.

As illustrated in Fig.4.7, route records in both *M-table* and *B-Table* are assigned to one of the following states: *VALID*, *INVALID*, *BROKEN*, and *DISCONNECTED*, according to the connectivity and responsiveness of neighbors, local interfaces, and destination networks. State *VALID* indicates the route record is a stable route. State *INVALID* specifies that the route record has expired, i.e., *timeout*

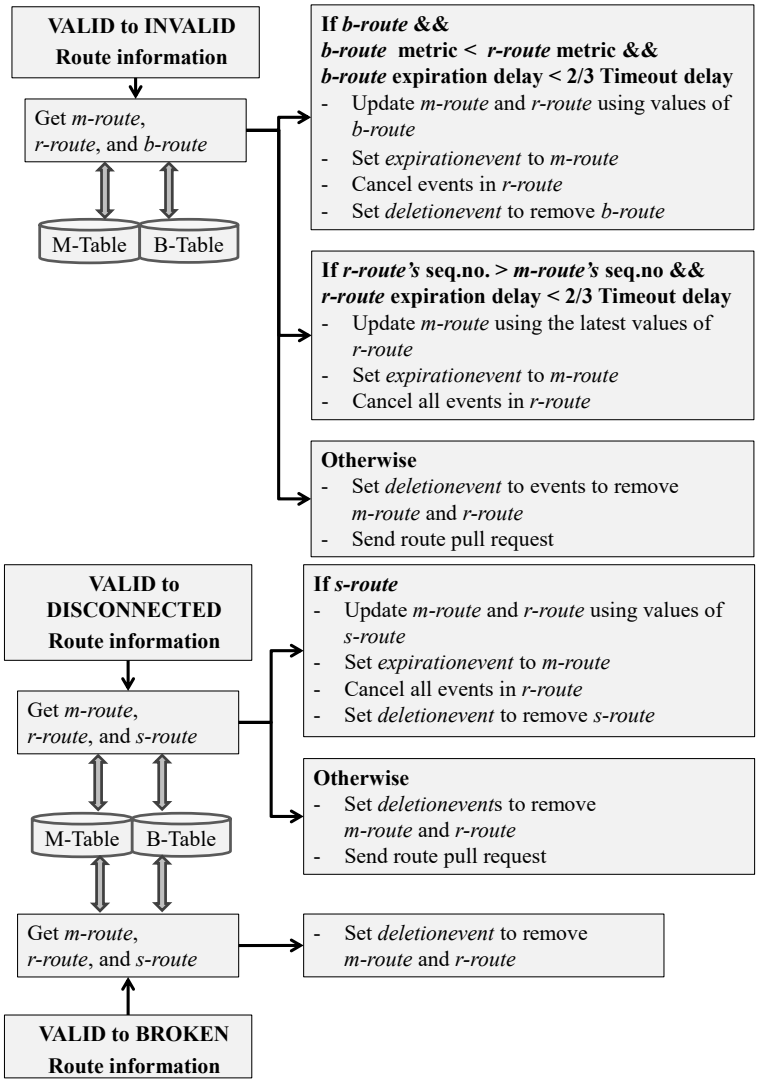


FIGURE 4.8: Method used in ESLR to invalidate route records

time expires. Routes are marked as *BROKEN* when the connected local interface or gateway router is unresponsive. Routes are moved to the *DISCONNECTED* state when the destination network is disconnected.

ESLR uses the route invalidation algorithm presented in Fig.4.8 to manage the route state change situations as follows:

- In case *m-route* is changed from *VALID* to *INVALID*, the nodes first check for a stable *b-route*. If a *b-route* is found and its cost is better than *r-route*, the nodes update both *m-route* and reference *r-route* using the information from *b-route*. Otherwise, by checking the sequence number from *r-route*, the nodes update *m-route* using the latest information from *r-route*. Else, the nodes remove both *m-route* and *r-route* from *M-Table* and send a route pull request among neighbors.

- In case *m-route* is changed from *VALID* to *DISCONNECTED*, and thus a *b-route* is found, the nodes update both *m-route* and *r-route* using the information from *b-route*. Otherwise, as presented in Fig.4.8, the nodes remove both *m-route* and *r-route* and send a route pull request among neighbors.
- In case *m-route* is changed from *VALID* to *BROKEN*, because the destination network is not responsive anymore, the nodes schedule an event to remove both *m-route* and *r-route* without sending a pull route request for the neighbors.
- If *b-route* is invalidated, despite the route state, an event is scheduled to remove the route.

4.3.2.2 Update route tables

The algorithm presented in Fig.4.9 is used to update the route records in both *M-Table* and *B-Table*. As previously explained, ESLR updates all receiving route records to its *B-Table* to maintain the stability of the topology table. That is, this method is the primary concern towards avoiding route oscillations. Therefore. As pointed out in Fig.4.9, nodes should update the routes in the *B-Table* as follow. If the received route is about a *r-route*, initially the node should update the route records by using the information listed in the *RUM*. The critical scenario is to update the route event information to automatic route management. That is the ESLR proposed to update the route expiration event based on the route metric, $\mathbb{T}_{ij}^{k}(t)$. ESLR use the following scenario to update the expiration event: if the calculated $\mathbb{T}_{ij}^{k}(t)$ is smaller than the cost of the existing *r-route*, the node should resume the current settling time after

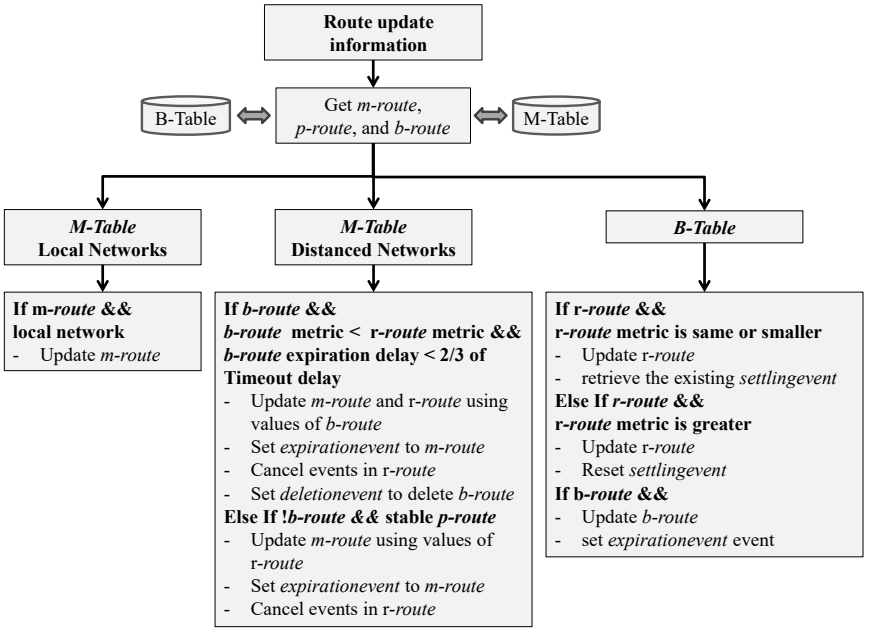


FIGURE 4.9: Method used in ESLR to update route records

updating the route attributes. Otherwise, settling time has to be reset. However, if the received route about a *b-routes*, regardless of $\mathbb{T}_{ij}^{k}(t)$, node should reset the expiration event.

Moreover, the algorithm presented in Fig.4.9 is further used to move those routes that are finishing their settling times at *B-Table* to *M-Table*. When *m-route* is updated, the nodes first check for a stable *b-route* with less delay than *r-route*. If found, the nodes then update both *m-route* and *r-route* using the values from *b-route*. Subsequently, the nodes mark *m-route* as changed, cancel the events associated with *r-route*, and schedule an event to remove *b-route* from *B-Table*. If no stable *b-route* is found, the nodes then update *m-route* according to the latest information from *r-route* followed by setting the changed flag of *m-route* and canceling the associated events of *r-route*.

4.3.2.3 Route advertisements

ESLR uses the data structures: ESLR advertisement header and route update message header (*RUM*) as coordinated messages. ESLR advertisement header and RUM header are presented in Fig.4.10 and Fig.4.11 respectively. ESLR use two types of route advertisements for stable and rigorous route broadcasting. The first type is periodic advertisements; periodic advertisement messages are generated at each *Periodic-Update* interval using the *VALID* routes in *M-Table*. The RUMs are added to the ESLR advertisement header according to the Maximum Transfer Unit (MTU) of a particular link. ESLR uses Eq.(4.11) to calculate the number of RUM that can be added to an advertisement header, where NoR

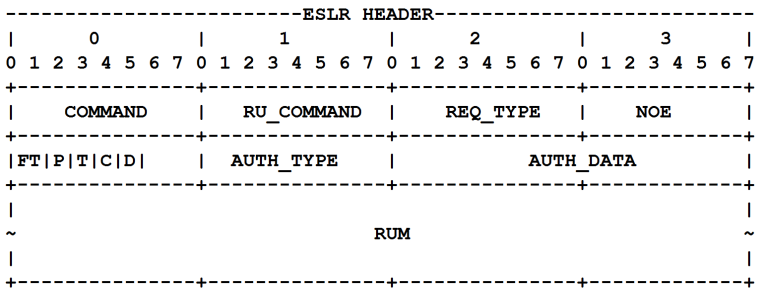


FIGURE 4.10: ESLR advertisement header

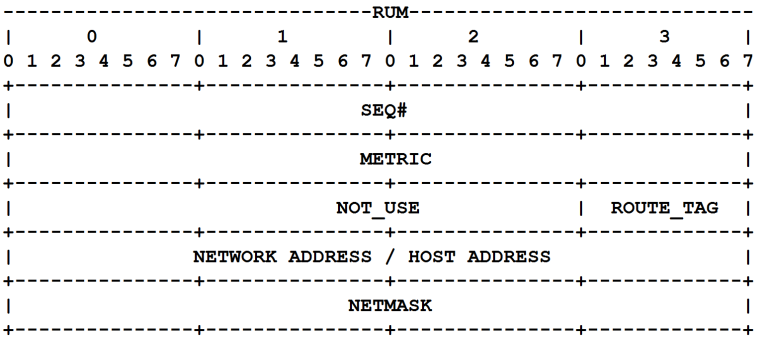


FIGURE 4.11: ESLR Route Update Message (RUM) header

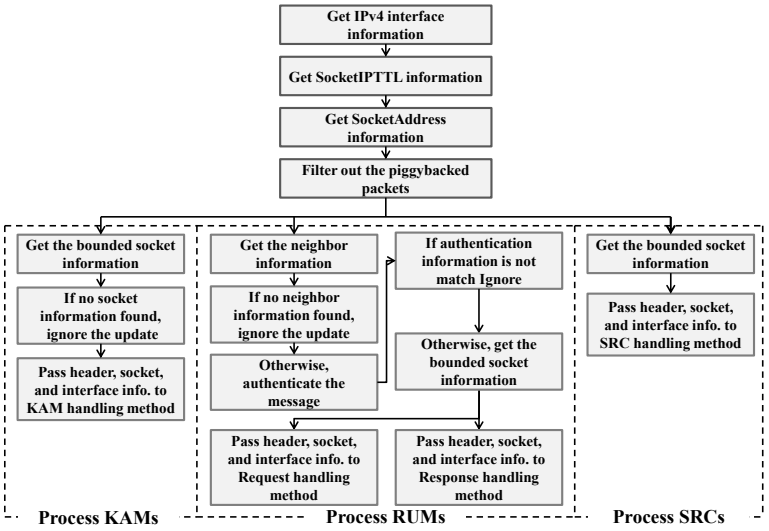


FIGURE 4.12: Method used in ESLR to process route advertisements

is number of records.

$$NoR = MTU - (sizeof(IPHeader) + sizeof(UDPHeader) + sizeof(ESLRHeader))/sizeof(RUM) \tag{4.11}$$

The second type is triggered advertisements; triggered advertisement messages are introduced to overcome the slow convergence problem of the Bellman-Ford algorithm. The triggered advertisements are further divided into two types: regular and fast-triggered advertisements. ESLR uses two provisions for regular triggered advertisement messages to avoid excessive loads created by the triggered advertisements in large topologies [140]. The first is to generate triggered advertisements only using the "changed" *VALID* routes in *M-Table*. The second is to maintain a time delay between two triggered advertisements. The delay, *TriggerHoldDown*, is an event that expires within a random interval between *TrigMAX* and *TrigMIN* (see Table 4.6). ESLR uses regular triggered advertisements to reduce the route convergence time of the Bellman-Ford algorithm to $O(|k| * |E|) * 5s$, where $|k|$ is the number of nodes, $|E|$ is the number of links, and 5s is the maximum time between two triggered updates.

It is notable that, even with the regular triggered advertisements, the route convergence time is quite high in complex networks [140]. Therefore, ESLR uses fast-triggered advertisement messages to manage critical scenarios, such as broken routes. Those advertisements do not assume *TriggerHoldDown*; nodes send fast-triggered update messages by setting *FT* and *D* flags in the ESLR advertisement header [141]. Such advertisements propagate through the network with a maximum of cumulative link propagation delay.

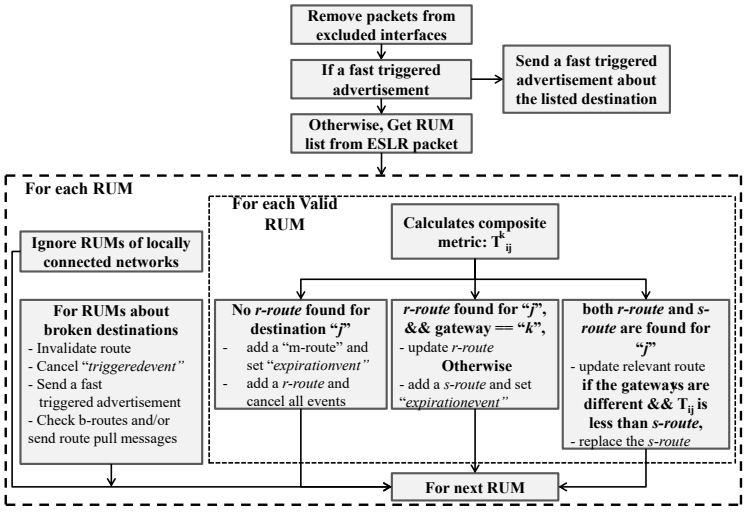


FIGURE 4.13: Method used in ESLR to process RUMs

4.3.2.4 Process route advertisements

Figure 4.12 shows the algorithm used by ESLR to process route advertisements. The algorithm is implemented to process one advertisement at a time, executes one event at a time because ESLR uses the diffusion computation concept to process route advertisements. As shown in Fig.4.12, when a node receives the advertisement message *RRQ*, *RRM*, or *SRC* the nodes validate the advertisement for authenticity. Finally, the node forwards the message to the relevant processing module. Nodes further validate each *RUM* carefully because *RRS* messages, i.e., *RUM*s can edit the routing table. As displayed in Fig.4.12, all advertisements received via excluded interfaces (piggybacked packets) are ignored. The remaining *RUM*s are authenticated using the credentials listed in *N-Table*. After validating *RUM*s, nodes process each *RUM* individually. In a situation where the $K1$ controllable coefficient value is set, servers advertise usage information at each *Svr-RtrTimer* period by filling the *SRC* header. Detailed information about the server router communication protocol and message processing method can be found in Chapter 5.

4.3.2.5 Process Route Update Messages (RUMs)

ESLR uses the algorithm presented in Fig.4.13 to processes *RUM*s. According to the algorithm, initially, nodes check for all poisoned route records listed in the advertisement, i.e., records on broken networks. If any poisoned route is found, the nodes execute the route invalidation process explained in Section 4.3.2.1. Then the nodes send fast-triggered advertisements among their neighbors to inform about the broken networks.

4.4 Route discovery and coordination activities

This section elaborates on the use of coordinated update messages to discover new routes and populate routing tables. Furthermore, this section explains the techniques used by ESLR to ensure loop-free routing by avoiding route oscillations, which is the main concern in the time-based routing paradigm.

4.4.1 Route discovery

For simplicity of understanding, let us suppose the scenario illustrated in Fig.4.14. As the figure shows, when *k2* receives a new destination network, i.e., *j*, from *k4*, because *k2* does not have *r-route* for *j*, *k2* adds *m-route* and a reference *r-route* for destination *j* by assuming *ep* is via *k4*. As shown in "stage 3" of Fig.4.14, when *k2* receives a new route from *k3* about destination *j*, given that the cost is greater than the existing *r-route*, *k2* maintains the newly received routes as the *b-route* for *j*. Subsequently, based on the cost, sequence number, and gateway of the receiving route records, *k2* updates the relevant route records and sets the associated events accordingly.

4.4.2 Loop-free routing with split horizon, poison reverse, and coordinated update messages

Since ESLR is designed based on the Bellmon-Ford algoriithm, the ESLR is open for the following limitations: route looping problem, slow convergence, and route osscilations. To avoid such limitations, dissertation proposes to use both split horizon and poisoned reverse techniques [75, 140] and speeds up the route convergence to ensure loop-free route computation. Suppose the simple scenario

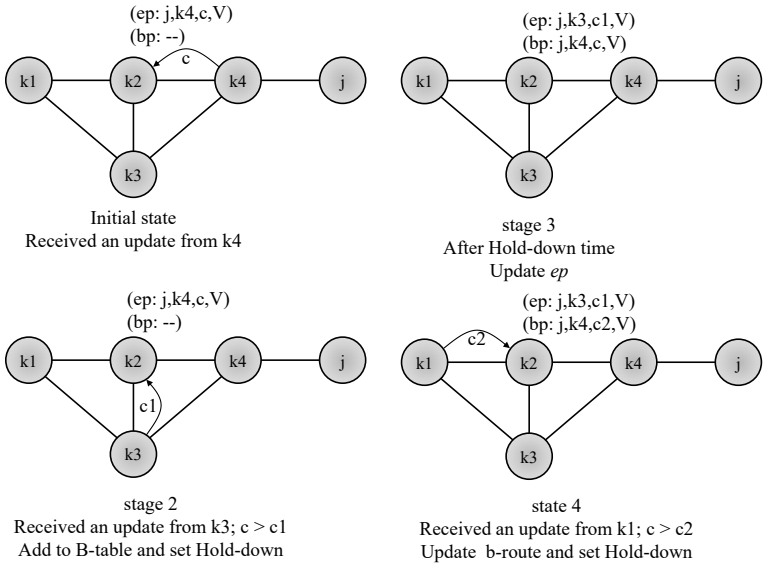


FIGURE 4.14: Route discovery in ESLR [format: (destination,gateway,delay,validity)]

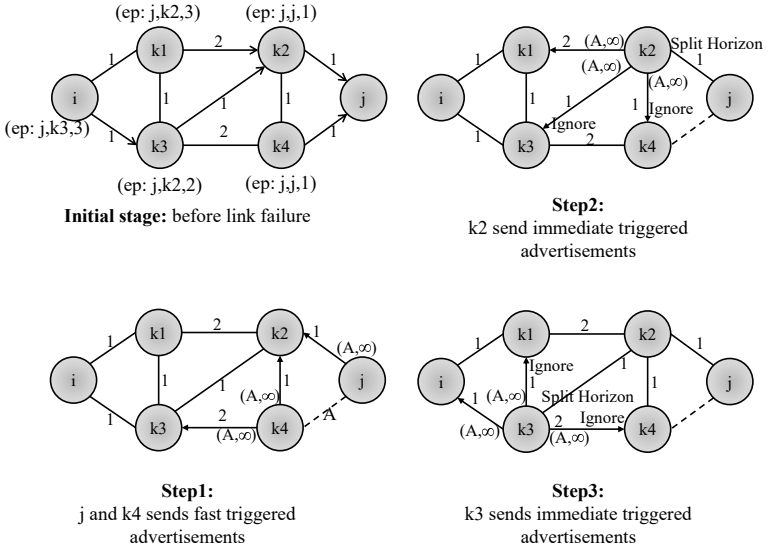


FIGURE 4.15: Message flow in link breakdown situation (dotted line shows disconnected link)

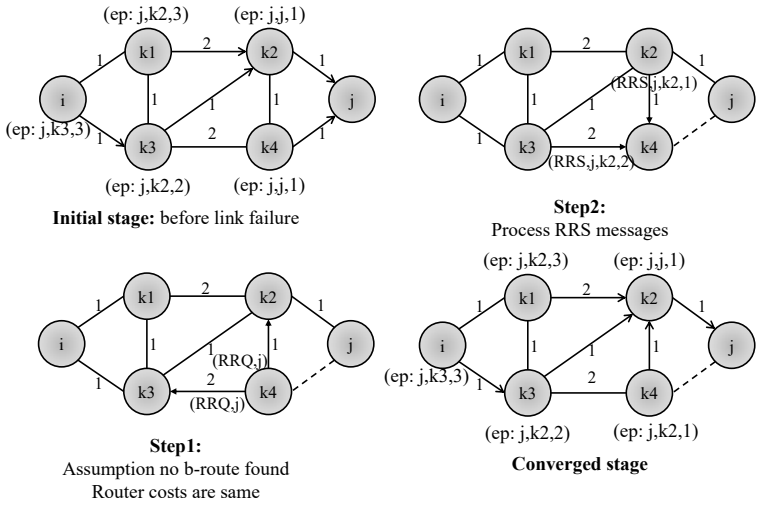


FIGURE 4.16: Coordinated messages used for link recovery (dotted line shows disconnected link)

illustrated in Fig.4.15 and assume that both split horizon and poison reverse techniques are configured. As illustrated in the figure, when the link between *k4* and *j* is broken, both *j* and *k4* send fast-triggered advertisement messages to their neighbors by marking the broken network as a poisoned route.

When *k2* receives an advertisement about the broken link, it also sends a fast-triggered update message about the broken link among its neighbors. However, because the split horizon is configured and *k2* learned the broken network from *j*, *k2* omits sending a triggered advertisement to *j*. Furthermore, when *k3* receives advertisement from *k2*, because *k2* has already received the information from *k4*, *k3* ignores the advertisement (see step 3 in Fig.4.15). This method helps prevent route looping and the count-to-infinity problem. As stated previously, for fast recovery in topology change situations, ESLR maintains *b-route* for each destination listed in *M-Table*. However, in large networks, it cannot be guaranteed that

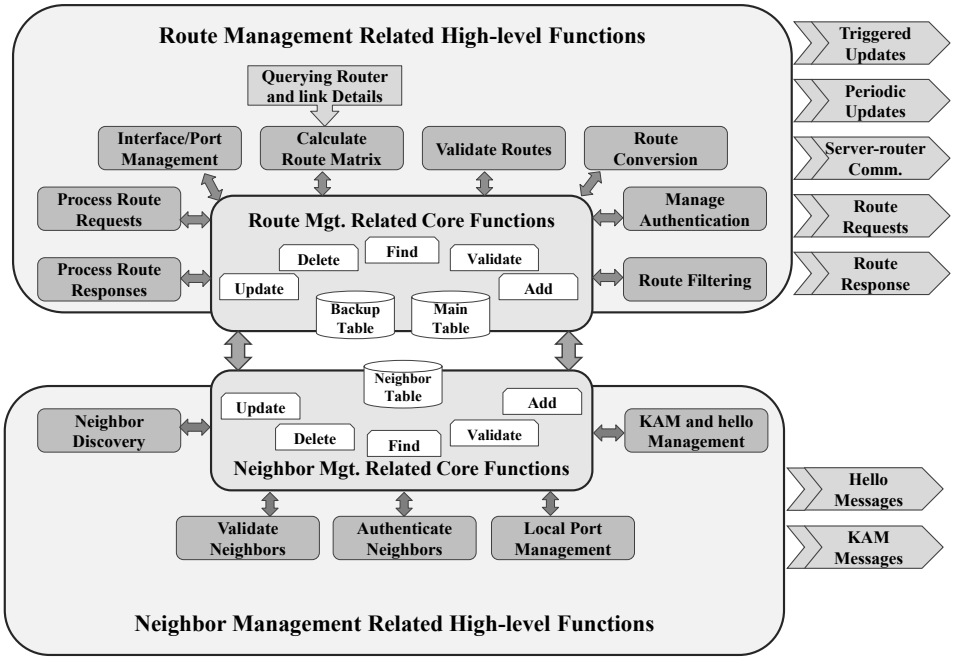


FIGURE 4.17: ns-3 ESLR module overview

the backup routes are always stable routes. Therefore, the route pull method was incorporated to ESLR for quick recovery in link failure situations.

Figure 4.16 displays the coordination activities among the neighboring nodes in a link failure situation. Assume the link between *k4* and *j* is broken and *k4* does not have a *b-route* for *j*. In this case, *k4* sends a fast-triggered update message to its neighbors followed by an *RRQ* message to request routes for the destinations affected by the disconnected link. When *k2* and *k3* receive the *RRQ* message from *k4*, both routers generate *RRS* messages by adding the requested networks. When *k4* receives the *RRS* messages from its neighbors, *k4* uses the algorithm presented in Fig.4.13 to recalculate the topology table.

Consequently, the methods explained above facilitate ESLR populate the routing tables without generating route oscillations, and thus provide fast recovery by avoiding routing loops in link failure situations. However, as an inherited limitation from distance vector algorithms, because ESLR has to send and receive a considerable amount of coordination messages between neighbor nodes, ESLR might display chattiness [125]. Theoretically, if the network links are low-speed, the chatty mode consumes a considerable amount of link bandwidth; chattiness is evaluated in Section 4.6.5.

4.5 Implementing the proposed ESLR protocol using ns-3

The first ESLR prototype is implemented using the ns-3 [121]. The ns-3 simulator is used because it supports most of the general packet structures, and its IP stack is similar to the Linux IP stack. The implemented protocol modules are illustrated in Fig.4.17. As the figure shows, both neighbor and

route modules are implemented with all supporting advertisement headers presented in Figs.4.4, 4.10, and 4.11, and functions to execute diffusion computation and coordinated advertisement methods. The protocol is implemented using object c++ language. *M-Table*, *B-Table*, and *N-Table* are implemented using key-value stores as on memory databases. The ns-3 event manager was used to initiate all timers, events, and the simulation test cases. The ESLR was implemented on a computer which is configured using Intel(R) Core(TM) i7-2600 CPU, 8GB RAM, and CentOS release 6.7 (Final). The source code of the latest version of ESLR is publically available and can be downloaded from.

4.5.1 Simulation topology information

Network topologies have a great impact on the performance of a routing protocol; selecting a proper ISP topology is mandatory. However, ISPs do not expose their router-level topology information because of the vulnerabilities to which ISP are exposed. Therefore, ISP makes their topology information publically unavailable. Alternatively, modeling techniques, such as BRITE and CAIDA, are used to model ISP networks. In addition, Rocketfuel uses publically available traceroute information to map the route level ISP topologies of various ISPs [133].

The ESLR is evaluated using two ISP topologies, Exodus-USA (AS3967) and Telstra-AUS (AS1221), presented in RocketFuel repositories [133]. Geolocation-based mapped backbone topology diagrams are depicted in Figs.4.18 and 4.19 (source: RocketFuel Maps [133]). Basic information about the selected topologies is given in Table 4.4. For experiment purposes, hundred traffic random generators were implemented and considered as clients. The clients were programmed to start and stop traffic

TABLE 4.4: ISP topology information [129, 133]

ASN No.	ASN1221	ASN3967
Name	Tesltra	Exodus
Proximity Factor	1.00	1–2
Degree	66	43
BB links	306	294
BB route degree (Max)	18	12
BB router degree (Agv.)	4–6	4–7
BB routers	108	79

TABLE 4.5: Link bandwidth allocation

	Number of times a link used for shortest path calculation	
Assigned bandwidth	ASN1221	ASN3967
1Gbps	1001 - 3000	301+
0.5Gbps	501 - 1000	101 - 300
0.1Gbps	1 - 500	1 - 100

TABLE 4.6: ESLR timers and default values

Timer	Description	Default value
Periodic-Update	Amount of time between periodic route advertisements	50s
Timeout	Amount of time after which a route is moved to INVALID state	100s
Hold-down	Amount of time route has to wait at the *B-able*	150s
KAMTimer	Amount of time between the two *KAM* advertisements	30s
NbrTimeout	Amount of time after which a neighbor is moved to INVALID state	50s
TrigMAX	Maximum time between triggered route advertisements	5s
TrigMIN	Minimum time between triggered route advertisements	1s
Garbage-collection	Amount of time that a route will be removed	10s
Svr-RtrTimer	Amount of time a server advertises to its gateway router	240s

generation in a timely manner creating burst traffic environments and generating flash crowd situations. Furthermore, by assuming streaming services, the servers were programmed to provide reply packets to each request they received. Clients and servers placed on the locations based on the Point of Presence (PoP) information [133], and those places are indicated by a "map pin" in Figs.4.18 and 4.19.

When configuring both topologies on the ns-3, the link delay and weight information presented in [133] were used to create nodes and corresponding network links among nodes. However, the Rocketfuel database does not provide link bandwidth information; link bandwidths were determined and assigned using the method. The Dijkstra's algorithm was ran at each node separately, and computed the shortest, i.e., minimal weight, paths (SPs) to the rest of the nodes in the topology. Then the number of times a link is used to generate SPs was counted. Finally, the bandwidth of the links was manually assigned to the selected links according to the number of times a link is used for SP computation. The link occurrence and assigned bandwidth information are presented in Table 4.5. Although EIGRP [142] is the closest protocol to ESLR, ESLR is compared against OSPF because EIGRP is not a prevailing routing protocol among ISPs. Furthermore, the Dijkstra's algorithm was used to compute SPs according to the Rocketfuel link weights and delays; computed SPs, i.e., cumulative propagation delay, are compared with *ep*s in order to ensure the accuracy of the computed network paths

4.6 Evaluate the proposed ESLR protocol using two ISP typologies

This section presents and discusses the obtained results to evaluate effectiveness of using network device state information for network path selection in ESLR. Consequently, the discussion can be divided as follows:

1. Comparison of route discovery time and accuracy of generated routes

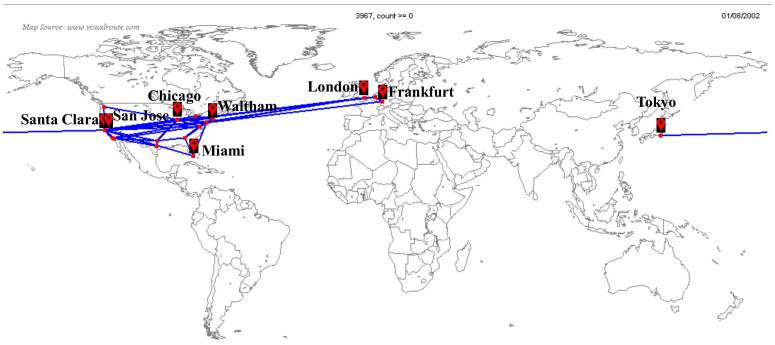


FIGURE 4.18: Exodus-USA (AS3967) topology

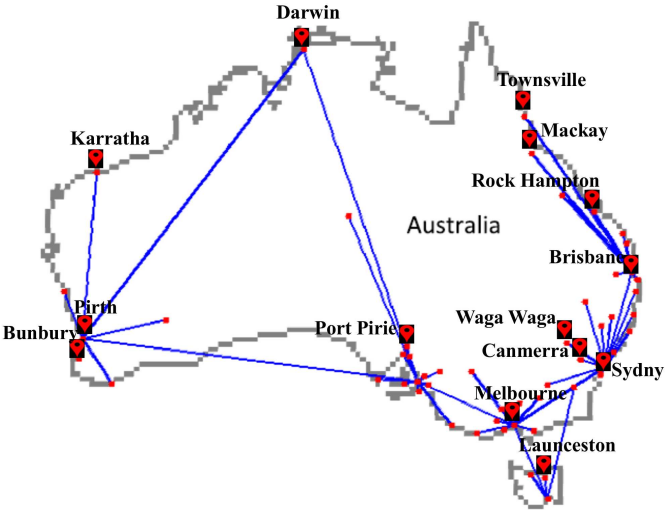


FIGURE 4.19: Telstra-AUS (AS1221) topology

2. Comparison of ESLR *ep*s with SPs, i.e., minimal delay paths, computed using Rocketfuel matrix
3. Comparison of end-to-end packet propagation delay with OSPF
4. Comparison of network resource utilization
5. Resource usage of ESLR for route computation and propagation.

Note that each client was started and let them download content from the servers to load both networks and servers at the beginning of each simulation. Furthermore, the parameters provided in Table 4.6 were used as the default parameters for this experiment.

4.6.1 Route discovery time and the accuracy of the ESLR routes

Initially, among the clients and servers placed in both topologies, several client-server pairs were selected based on PoP density and geographical displacement. The selected client-server pairs in ASN3967 and ASN 1221 are presented in Tables 4.6 and 4.7 respectively. The route discovery time is

TABLE 4.7: Route discovery time (ASN3967)

	Route discovery time (s)	
	ESLR	OSPF
Santa Clara,CA444 – Tokyo526	2	58
San Jose,CA471 – London275	4	56
Tokyo525 – London276	119	62
London276 – Frankfurt184	4	57
San Jose,CA460 – Waltham,MA569	75	58
Frankfurt184 – Tokyo525	76	62
Chicago,IL155 – Miami,FL285	45	58

TABLE 4.8: Route discovery time (ASN1221)

	Route discovery time (s)	
	ESLR	OSPF
Townsville,Aus.4282 – Launceston,Taz.1867	107	68
Brisbane,Aus.1800 – Karratha,Aus.1865	123	61
Darwin,Aus.1837 – Sydney,Aus.4241	75	61
Darwin,Aus.1838 – Rockham.,Aus.4195	111	62
Perth,Aus.4160 – Canberra,Aus.1814	115	57
Perth,Aus.4161 – WaggaWagga,Aus.4291	112	62
Mackay,Aus.3851 – Bunbury,Aus.1804	166	67

measured in the first experiment. That is, the time that a client has to wait until it receives the route record of its partner server (route discovery time) and vice versa were measured. The results obtained are presented in Tables 4.6 and 4.7. After both clients and servers had discovered the route records, a PING request was sent from clients to its partner servers, and number of hops in the shortest path was measured as the next experiment. The results obtained are presented in Tables 4.8 and 4.9.

The following discussion uses the results presented in Tables 4.7, 4.8, 4.9, and 4.10. Note that RF stands for Rocketfuel. As given in Tables 4.7 and 4.8, the protocol initialization time for ESLR is proportional to the distance between the destinations (number of hops). This means that ESLR displays the inherited limitations of the distance vector algorithms. However, protocol-initialization time is usually maintained at less than 1.5 factor of the OSPF route discovery time because ESLR uses triggered advertisements for route distribution. Moreover, ESLR can discover all possible network paths in all nodes in an average of four periodic update cycles using the parameters presented in Table 4.6.

Furthermore, according to Tables 4.9 and 4.10, all three routing paradigms provided a virtually similar number of hops in their shortest paths. However, it is notable that the hop count for *ep*s vary while the hop count for SPs provided by OSPF and RocketFuel is constant. The reason for this behavior is that ESLR dynamically adjusts its network paths according to the states of the network device. Thus, a number of hops in *ep* might change according to the selected network path.

TABLE 4.9: Average number of hops(ASN3967)

	Number of hops		
	ESLR	OSPF	RF
Santa Clara,CA444 – Tokyo526	2-4	2	3
San Jose,CA471 – London275	2-4	3	5
Tokyo525 – London276	6-8	6	7
London276 – Frankfurt184	2-4	2	3
San Jose,CA460 – Waltham,MA569	7-11	6	9
Frankfurt184 – Tokyo525	8-11	7	9
Chicago,IL155 – Miami,FL285	5-8	4	6

TABLE 4.10: Average number of hops (ASN1221)

	Number of hops		
	ESLR	OSPF	RF
Townsville,Aus.4282 – Launceston,Taz.1867	8-11	7	9
Brisbane,Aus.1800 – Karratha,Aus.1865	8-11	7	8
Darwin,Aus.1837 – Sydney,Aus.4241	4-7	4	5
Darwin,Aus.1838 – Rockham.,Aus.4195	8-12	7	8
Perth,Aus.4160 – Canberra,Aus.1814	6-9	6	7
Perth,Aus.4161 – WaggaWagga,Aus.4291	6-9	5	6
Mackay,Aus.3851 – Bunbury,Aus.1804	7-11	7	8

4.6.2 Minimal delay network path selection

Mahajan et al. inferred that link weights calculated while generating the RocketFuel topologies represent the most realistic and accurate route selection [143]. Therefore, the propagation delay of *eps* was compared with that of the shortest paths generated using Rocketfuel link weights in order to compare the accuracy of the *ep*s calculated by ESLR. The results obtained are presented in Figs.4.20 and 4.21. The ESLR *CCV*s were configured as $K1$=0, $K2$=1, and $K3$=1 on this experiment for a fair evaluation.

By comparing the graphs presented in both figures, it can be observed that 80% of the time, ESLR can use the network device state information to provide minimal delay network paths compared with the SPs provided by Rocketfuel metrics. However, it is notable that when the selected destinations are near, i.e., the top middle graph of Fig.4.20, both Rocketfuel and ESLR select the same path, and thus the delays of *ep*s are slightly diverse. The reason is that the *ep*s represent the cumulative delay of the router, and the links between the destinations, and the delay is given by the RF method is the cumulative delay of the links. Therefore, the delays given by ESLR shortest path is quite high. However, when the destinations are far away from each other, ESLR can calculate the most minimal delay compared to the RF because the ESLR select the network path by considering the traffic matrix; ESLR avoids the congested links and routers effectively. This proves that the ISPs should optimize their IGPs according to the traffic matrix. Furthermore, this scenario explains that the basic method, which is contemplating the influence of traffic matrix by analyzing the network devices is possibly an effective method for TE.

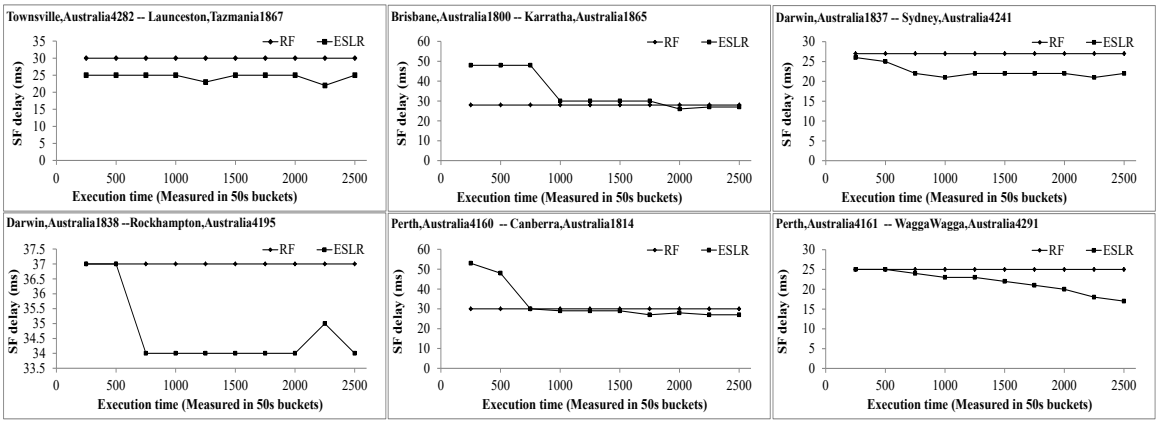


FIGURE 4.20: Shortest path delay comparison (ASN1221)

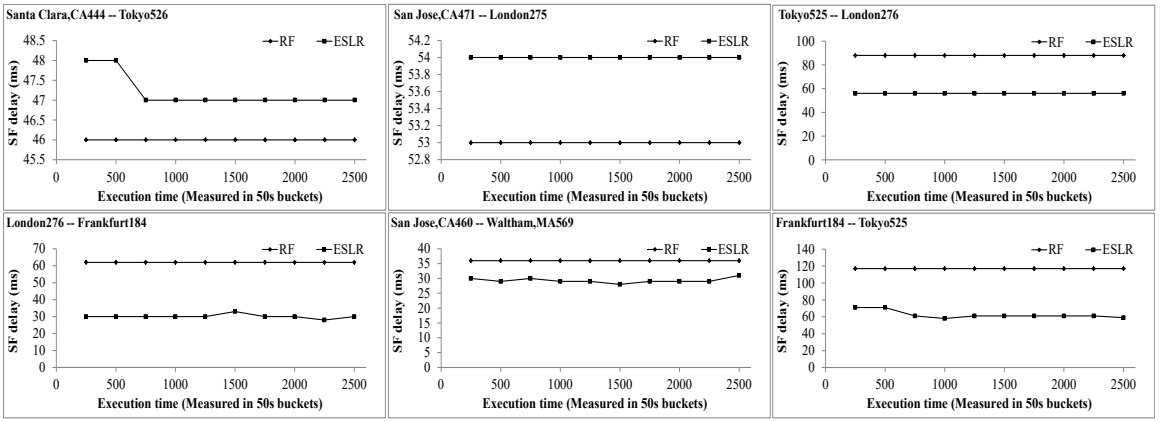


FIGURE 4.21: Shortest path delay comparison (ASN3967)

4.6.3 End-to-end propagation delay

The end-to-end propagation delay is analyzed by measuring the round trip time (RTT) of the reply packets at each client presented in Tables 4.7 and 4.8. For this experiment, the clients were programmed randomly to change the IP address of the destinations in the middle of the simulation; the motivations is to analyze the impact of the dynamic redirection of the clients for network devices, and hence network path selection. In addition, the ESLR *CCV*s was configured as follows: $K1$=1, $K2$=1, and $K3$=1, to study the impact of the content servers for network path selection. The measured results are presented in Figs.4.22 and 4.23.

It can be observed that the *ep*s provided by ESLR reduce the propagation delay by 20%-30% compared with OSPF by analyzing the graphs presented in both figures. The reason is that ESLR calculates the minimal delay paths according to the states of the networks. Nevertheless, graphs presented in Figs.4.20 and 4.21 showed that more than 80% times ESLR was able to provide the minimal delay paths. Therefore, 80% of the times the packets always use the paths through less congested routers and links. This further proves that the ESLR effectively reduces the end-to-end propagation delay by adapting to the dynamic behavior of the network devices. ISPs can use this behavior of ESLR

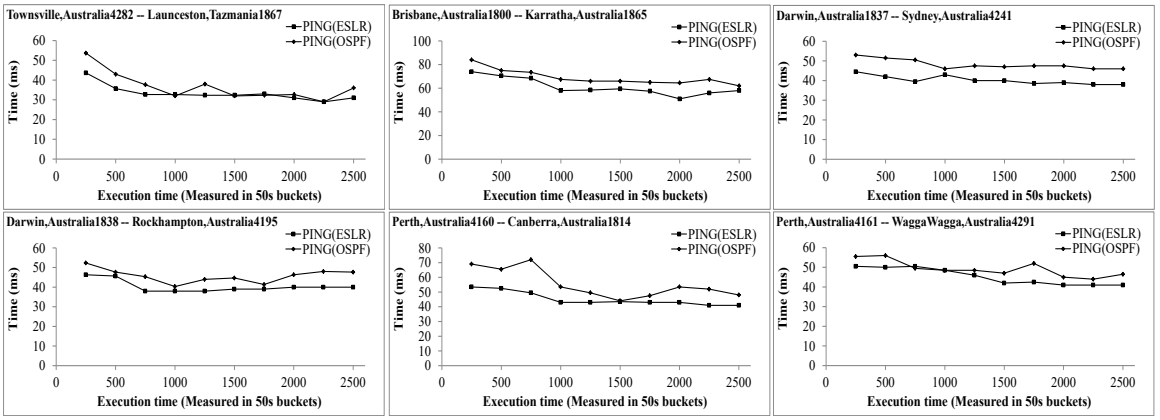


FIGURE 4.22: End-to-end packet propagation delay (ASN1221)

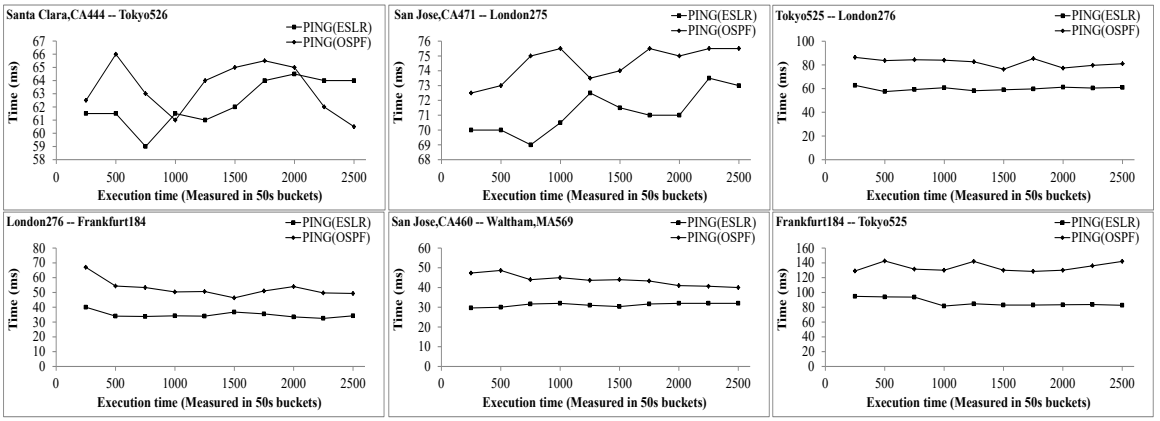


FIGURE 4.23: End-to-end packet propagation delay (ASN3967)

to observe the dynamicity of the network devices, and thus, route their subscribers by avoiding flash crowd situations. Furthermore, as stated in [126] and depicted in Fig.4.1, CPs can use the ESLR route matrix to generate relevant end-user redirection policies because the route metric is calculated using a universal metric, time. Hence, this can be a possible business model that the current NSPs, i.e., ISP-CP collaboration, can use to guarantee that their subscribers are redirected to the best server via the minimal delay path (See Fig.4.1).

However, according to the results presented in Figs.4.22 and 4.23, ESLR displays a slightly higher propagation time at the beginning of the simulation, and it is stabilized after 8–10 update cycles. One reason for such behavior is that at the beginning of the route discovery process, a node always uses the first route it received for a particular destination as ep. This route might not be the minimal delay path to the destination. However, ESLR uses this approach to overcome the slow convergence process of the Bellman-Ford algorithm. The other reason is that ESLR uses a considerable amount of coordinated messages *hello*, *RRQ*, and *RRS* for the ESLR initialization process. Therefore, ESLR chattiness is high at the beginning of the route discovery process.

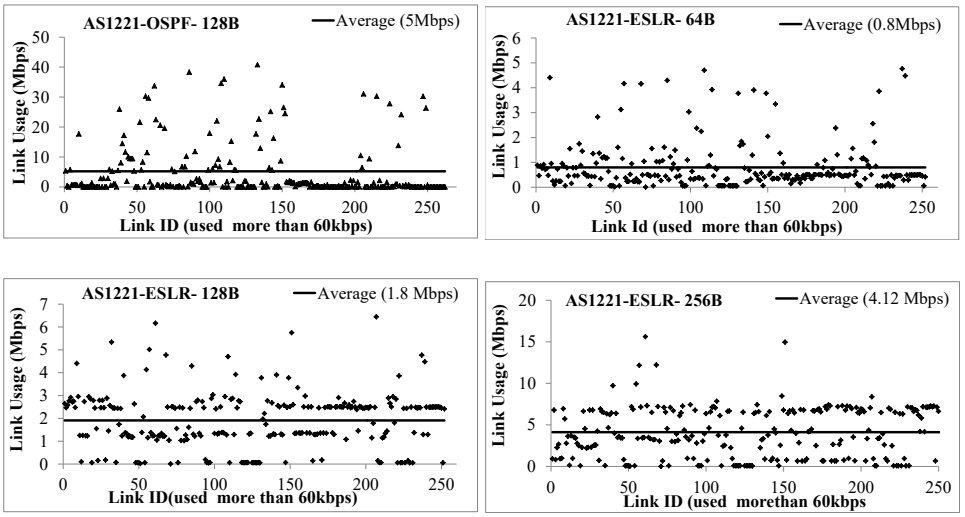


FIGURE 4.24: Network resource utilization (ASN1221)

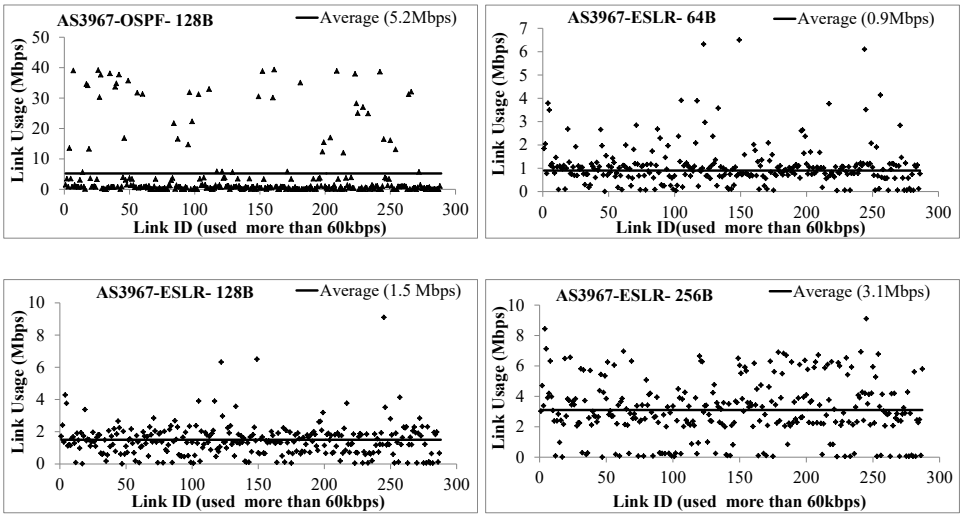


FIGURE 4.25: Network resource utilization (ASN1221)

4.6.4 Network resource utilization

Series of experiments were conducted to measure the link utilization by setting the average request and reply packet sizes to 64 B, 128 B, and 256 B. All clients were configured to download data from the intended content servers for this experiment. Moreover, the clients are programmed to change the content servers' IP addresses randomly in the middle of the simulation. Furthermore, the Rocketfuel link matrix values were configured as the OSPF link matrix for shortest path selection. Note that the link utilization was measured by computing the fraction of the links used for data delivery as a one-way-based value of the link' capacities. The results obtained are presented in Figs.4.24 and 4.25.

As shown in these figures, OSPF utilizes several links on the topology while leaving other links unutilized at the same time, as stated in Chapter 2. and [73]. However, ESLR utilizes approximately 90% of

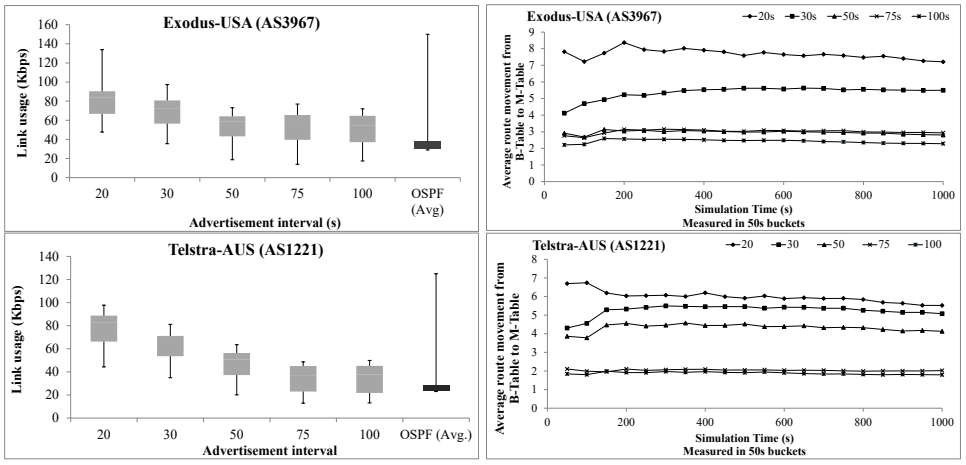


FIGURE 4.26: Link occupancy for route management

the links on both topologies. It is also notable that none of the links display excessive loads compared with OSPF, i.e., 50Mbps. The reason is that ESLR selects network paths based on propagation delay, and the propagation delay is thus calculated mainly according to the states of the network resources. Strategically, when a node identifies that a network path (i.e., a route) is busy, and the propagation time is higher, the node then switches the main route with the backup route. It is then proven that the proposed method can successfully contemplate the network device state information for effective network path selection by minimizing network congestion and maximizing network resource utilization.

4.6.5 Network resource usage for route management

Several experiments were conducted to analyze the resource usage of ESLR for its route computation and propagation. In particular, the fraction of the links and routers used by ESLR for coordinated activities was measured. For accurate and proper evaluation, in each experiment, the ESLR default values presented in Table 4.6 are modified according to the *Periodic-Update* interval as follows: *Time-out* time is configured as twice *Periodic-Update*, and *Hold-down* time is configured as three times the *Periodic-Update* interval.

The first experiment was conducted to measure the link usage by configuring different route advertisement intervals: 20 s, 30 s, 50 s, 75 s, and 100 s, and respective timer values in order to study the most appropriate timer values. For this experiment, all coordination activity messages, triggered advertisements, and periodic advertisements were considered to evaluate the link usage of ESLR. The results obtained are presented in Fig.4.26; the results are calculated by computing the average link usage of all links in a topology. According to the figure, link usage is comparatively high when ESLR uses small advertisement intervals, e.g., 20 s, and low when ESLR uses big advertisement intervals, e.g., 75 s. However, in either case, link usage is 20 Kbps to 30 Kbps higher compared with OSPF link usage. First reason is that ESLR still does not support route summarization, i.e., VLSM, and therefore, the

TABLE 4.11: Router usage for coordinated activities on Exodus-USA (AS3967)

	ESLR behaviour	
Degree of the router	Average packet size	Router usage
1-3	142B	250Kbps
4-7 (Avg.)	168B	323Kbps
8-10	151B	300Kbps
10+	147B	291Kbps

TABLE 4.12: Router usage for coordinated activities on Telstra-AUS (AS1221)

	ESLR behaviour	
Degree of the router	Average packet size	Router usage
1-3	138B	250Kbps
4-6 (Avg.)	159B	530Kbps
7-10	154B	456Kbps
10+	96B	291Kbps

average advertisement packet size is high. The other reason is the chattiness caused by the coordinated activities that are used for loop-free route computation.

The next experiment was conducted to analyze the router behavior according to the degree of the router, i.e., the number of neighbors and the results obtained are presented in Tables 4.11 and 4.12. According to both tables, the average ESLR advertisement packet size is 150 Bytes on both topologies. The reason is that the number of backbone links in both topologies is almost the same, and hence, the average number of *RUM*s contained in an advertisement packet is also similar in both topologies. Furthermore, it is notable there is a variance of router usage according to the degree of the router. The reason is that nodes should process advertisements received from their neighbors; the resource usage for processing route advertisement messages is proportional to the number of neighbors. In addition, by analyzing Tables 4.11 and 4.12, it can be observed that router usage is high when the degree of the router is on the average (see Table 4.4). The reason is that ESLR strategically uses the routers with the average degree of neighbors to calculate ep.

The final experiment of this series was to measure ESLR for route changing frequency because ESLR is developed as a delay-based routing protocol, and thus the metric "delay" is a highly varying parameter. In this experiment, the number of routes being updated or replaced in *M-Table* at each *Hold-down* time event was measured by changing the route advertisement intervals and the other respective timer values. The obtained results are presented in Fig.4.26. According to the figure, ESLR can maintain an average of three updates for 200 s when it uses 50 s as the route advertisement interval. The tradeoff is that small advertisement intervals cause a high burden to both links and routers, but help fast router recovery in link failures situations regardless of *b-route*s. On the contrary, big advertisement intervals create less of a burden to both links and routers, but the link recovery and route propagation times are high. However, given that ESLR maintains *b-route*s, ESLR can recover link failures at the milliseconds scale.

Chapter 5

Use SoR and ESLR to enhance content navigation

Transport requires navigation. Navigation is the key principle of finding a destination and travel towards it. Travel is erratic and, mostly uncertain until a path towards the destination is found and known. Given the Internet architectures, constitute of clientserver communication and the servers are placed in remote locations, clients should navigate their data request across the Internet and reach the intended server. Similarly, servers should navigate the required content across the Internet to make sure the content have been reached to the intended user. As pointed out in Chapter 1, the content navigation problem can be divided into two main parts:

1. Select service points
2. Select network paths to reach the selected service point (a detailed explanation was given in Chapter 4)

Consequently, this chapter is mainly focusing on proposing a foreseeable solution to solve the service point selection problem for effective content navigation.

Generally, in WWW, content providers solve the service point selection problem; web hosts use Domain Name System (DNS) to let the client know about the service points. ISPs address the path of network selection problem; ISPs determine the paths, which can be used to reach the service points using a traditional routing protocol such as "shortest path routing" protocols. However, as depicted in Fig.5.1, with the introduction of content delivery networks (CDNs), the content navigation problem became more complicated. As a matter of fact, several new questions were emphasized:

1. Where to place the content?
2. How to select the nearest service point based on user location?

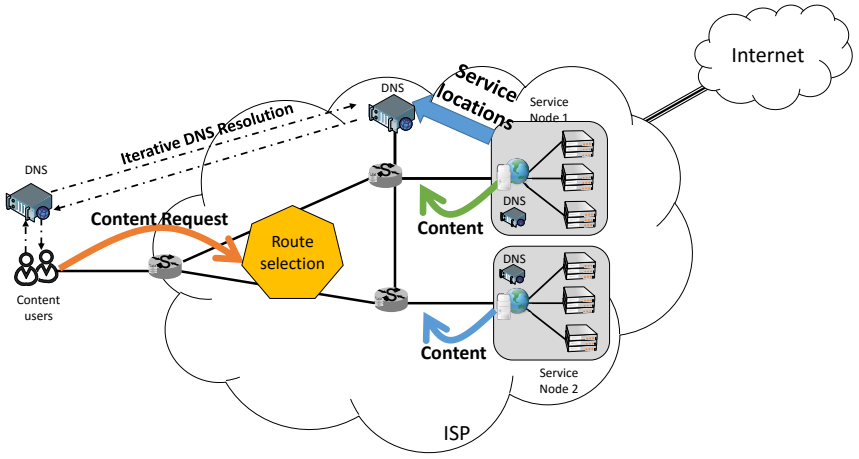


FIGURE 5.1: CDNs place at the edge of the Internet

3. How to notify the users about the selected service points?

4. How to select network paths according to the new content placement?

Most importantly, the question, "are existing technologies such as DNS and shortest path routing protocols capable enough to solve the content navigation problem in the modern CDNs," became the utterly significant consideration of researchers, developers, and business bodies.

To this end, CDNs used modified DNS architectures, such as FreeFlow DNS [18, 59], to solve the service point selection problem [14, 53] (Request routing, RR). Despite the advantages of the DNS-based RR, when CDNs adhere to the DNS and select service points to their subscribers, as explained Chapter 2 Subsection 2.2, several limitations degrade the performances of both the CDNs and users. The problems can be pointed out as follows:

1. Use very short TTL values to update the redirection policies [14]

2. No standard or proper methods to calculate TTL values according to the server state [14]

3. Misjudge user location when determining the nearest service point [14]

4. Users are required to wait until TTL expires to get the latest information about the service points [14, 53]

ISPs, on the other hand, as displayed in Fig.5.1, blindly route huge volumes of traffic introduced by CDNs without considering the traffic matrix of CDNs as a parameter for network path selection [73, 82]. This phenomenon degrades the performance of network path selection and network resource optimization. Hence, ISPs are attempting to optimize network path selection, and eventually, network resource utilization by using routing protocols, which are optimized for single arbitrary metric, shortest path selection. However, [73] anticipates that the network resource utilization, i.e., network path

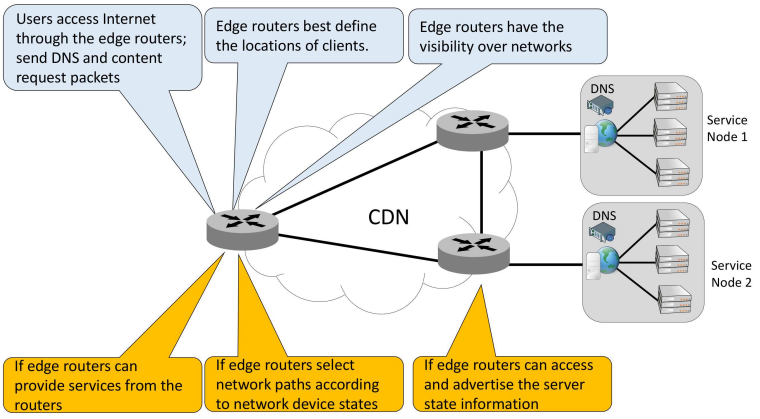


FIGURE 5.2: Effectiveness of gateway routers as an intermediary for ISP CDN collaboration

selection, is highly contingent of the traffic matrix leased by the content providers. Therefore, the effectiveness of solving the content navigation problem highly depends on an active collaboration between the ISPs and CDNs. Alternatively, the dissertation proposes a method to use edge routers of CDNs to, create a foreseeable collaboration between ISPs and CDNs, and use the collaboration to solve the content navigation problem.

5.1 Placing SoRs as the edge routers in CDNs

Despite the advantage of conventional routers, the routers are a lack of packet analyzing capability and in-network storage modules. In fact, to use the gateway routers as the intermediary of the ISP CDN collaboration, the gateway routers should satisfy several factors. As illustrated in Fig.5.2, the gateway routers should comply with following features:

1. Gateway routers should have a method to access the CDN server state information
2. Gateway routers should have a method to advertise the server state information among other routers
3. The routers should have a method to find the network paths according to the network device state information (this is already discussed in Chapter 4)
4. Gateway routers must be capable of providing user services

Thus, as explained in Chapter 3, SoRs have the capability of analyzing packets from IP layer through Application layer, and SoRs have in-network storage modules to store the necessary information. Therefore, as depicted in Fig.5.3, the dissertation proposes to place SoRs as the gateway routers of both users and the CPs. Thereby, the dissertation is proposing to select the gateway routers as the intermediary of the ISP and CDN collaboration for the following reasons:

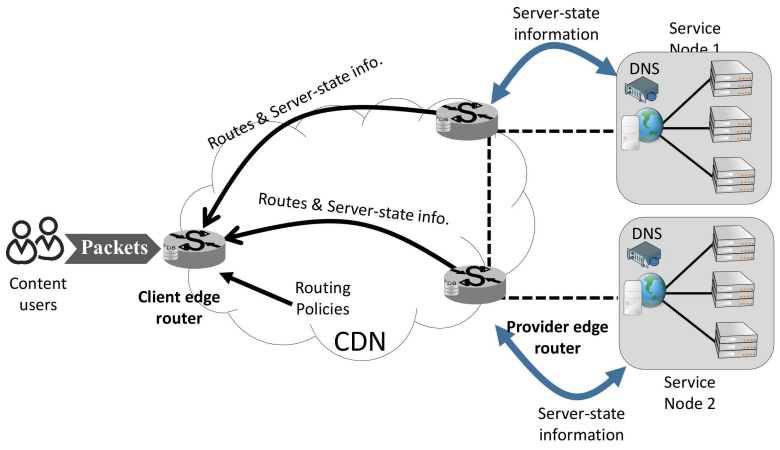


FIGURE 5.3: Place SoRs at the edges of CDNs (dotted lines indicate the communication between service nodes to share necessary control information)

1. User gateway routers are the best place to define the location of the end users because end users access CDN or the Internet using their gateway routers.

2. Both user and content providers' gateway routers experience the traffic matrix of the content providers, and thus, routers can use the traffic matrix as a parameter to solve the network path selection problem.

3. Gateway routers exchange the network topology changes in the order of seconds and can use that information to determine the best paths to navigate content through the network.

4. User gateway routers can be used to intercept DNS messages within one hop distance, and to leverage the server selection using to both traffic matrix and network route matrix as the parameters to solve the server selection problem.

5. Content providers can use their gateway routers to notify redirection policies without adhering the limitations of the DNS TTL values. Consequently, this chapter discusses the notion of introducing SoRs as the gateway routers to work as an intermediary to for foreseeable ISP and CDN collaboration.

5.2 Collect and distribute network and server state information

In general, ISP network-state information is useful in eliminating bottlenecks and avoiding flash crowds, whereas CDN server state information is helpful in determining the least-busy servers [14, 26, 73]. In essence, ISPs and CDNs collaboratively can use such information to select best servers to their subscribers, and thus, forward the subscribers to the selected server using the least busy network paths. Similarly, in [27], authors incurred that, with given necessary information, the content navigation delay, particularly server selection delay, can be reduced to the order of milliseconds. Hence, as illustrated in

Fig.5.3, the SoRs are strategically placed at both user and content providers edges to collect and share both ISP network-state and CDN server state information among the other SoRs in the CDN.

SoRs collects the ISP network-state information by accessing the routing tables; this study assumes the ISPs are configured with ESLR which is explained in Chapter 4. As discussed in Chapter 4, the ESLR is designed to use network device state information, i.e., processing and propagation delay, to effectively select network paths and thereby enhance the network path selection and network resource utilization. This means that the SoRs can get the up-to-date network state information by accessing the routing table of ESLR. Moreover, as already stated in Chapters 1 and 2, route advertisements carry the network information among the network routers, and the routing protocols uses numerous technologies to speed-up the advertisement propagation. This feature of the routing protocls is used in the proposed collaborative method to collect and advertise server state information among the SoRs. Subsequently the dissertation proposes an extension to the ESLR to collect and advertise the server state information. CDN content servers advertise the necessary server state information to the provider's gateway routers, i.e., SoRs using that extension. Consequently, working as an intermediary, SoRs collect and advertise the required server information and network topology information. This concept is factually similar to the reverse proxy technique [14], which is the core building block of CDNs.

The SoRs then uses the collected information to leverage DNS-based CDN redirection by addressing the server selection problem. In essence, SoRs use the gathered information to reduce the delay caused by hierarchical DNS server resolution process, which is a fundamental limitation of DNS-based CDN RR discussed in [20, 59]. Thereby, SoRs attempt to accelerate the connection initiation process, and thus, adapt the content server changes on small time scales. Furthermore, as explained in Chapter 3, the SoRs possess the capability to analyzing Application-layer data of packet streams and take necessary actions based on that information. Subsequently, this feature of SoR is used by the dissertation to provide on-the-fly content-aware packet redirection, which redirects packets by analyzing application layer information, i.e., information provided in unified resource locators (URLs). This concept is apparently similar to the interception proxy concept [14], which is yet another essential building block of modern CDNs.

5.2.1 Network state information

Network state information defines the current condition of the network; congestion and availability of network routers and links or flash crowd situations. As a matter of fact, network management protocols monitor the networks and adjust the network routing accordingly. Routing can be summarized as the process that routers decide how to forward datagrams based on its destination address by comparing the network states information that router keeps in a special entity known as routing tables. Routing tables contain entries for each network the router can reach, what is the adjacent node the router should

TABLE 5.1: An instance of the ESLR routing table

Destination IP	Mask	Gateway	Local interface	Sequence#	Validity	Metric (delay)	Control information
192.168.10.0	255.255.555.0	10.0.1.1	1	12	Valid	150ms	Main
192.168.10.0	255.255.555.0	10.0.5.4	3	18	Valid	180ms	Backup

contact to reach each destination network, the cost of reaching the next hop, and the validity of the next hop.

However, as pointed out in Chapter 2, the existing interior gateway routing protocols alone does not necessarily consider the dynamic behavior of network devices for network path selection [73, 82]. Therefore, the dissertation assumes that the CDN networks are configured with ESLR. As depicted in Fig.4.1, ESLR proposed a hypothesis for using network device state information for network path selection [96]. The ESLR proposes a method for converting network device state information into a standard scale, time. Thereby, the ESLR selects an effective path (ep), which is the minimal delay path between two destinations, as the best route. ESLR calculates the ep by measuring the packet processing delay of routers and the packet propagation delays of the links. Furthermore, if necessary, CDNs can use the packet processing delays servers as a parameter to calculate the ep.

It is a fact that ESLR contemplates the influence of the content providers' traffic matrix by measuring both routers and the links for the packet propagation delay even if the servers are not a parameter of the metric calculation. Therefore, ESLR routing table ideally maintains the minimal delay path consists of less busy routers and less congested links to reach each destination. Besides, as presented in Fig.4.1, ESLR provides an interface to access the routing table information with the motivation of providing effective user services. In essence, ESLR routing tables contains the up-to-date network state information. Therefore, the SoRs can use the ESLR routing table to collect the up-to-date network information feasibly. Consequently, gateway routers, i.e., SoRs, determine the network state information using the ESLR; SoRs solve the "network path selection problem" of "content navigation" using the ESLR. This redefines the service location selection problem; the best place might not be nearest service point, thus the server which can be reached using minimal delay paths.

Table 5.1 displays an instance of the ELSR routing table. According to the table, ESLR provides the destination details, next adjacent node to reach towards the destination, the delay to reach the destination, last updated time or the expiration time, and the validity of the record. Moreover, as presented in the table, ESLR provides a backup route to use in case of sudden failures of the main route. Such features help the SoRs to select the network paths adequately and use those network paths to select service locations to the users collaboratively.

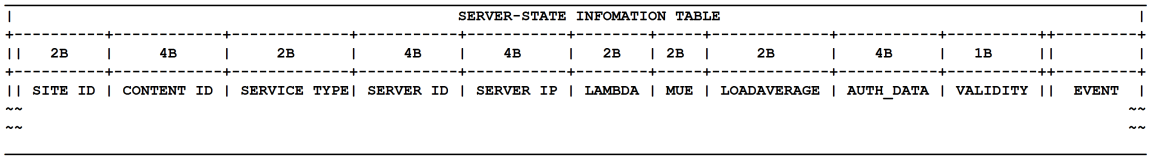


FIGURE 5.4: Table structure used by SoRs to store server state information

TABLE 5.2: Proximity approximation methods

Method	Description
Reactive probing	This method uses a special DNS server to reactively probe the customers prior to connection initiation. Then determine the nearest service point.
Proactive probing	This method uses special DNS servers to measure the delay between the service point and the local DNS server. This method assumes that the Local DNS server is always locate with in the local network. This is the commonly used method in CDNs.
Connection monitoring	This method is required the content servers to periodically advertise the server state information to a special DNS server known as Auth. DNS server. Servers periodically advertise the RTT information to the corresponding Auth. DNS servers. Auth. DNS servers maintain a huge database to store the of RTT from each server. Customers are redirected to the service point based on the computed delay.

TABLE 5.3: Assumption used to assign TTL values of DNS records [14]

RTT	TTL
Unknown	Very Short value
Low	Long value
Medium	Medium Value
High	Short value

5.2.2 Server state information

There is limited information available to answer the questions "how CDN management servers or forward proxy servers determine the best service location and, how to create DNS records according to the server states?" References, [14] and [20] give some hints about several high-level functional descriptions related to service point selection problem. Furthermore, [14] and [20] provide how CDN uses DNS to notify users about the services points, i.e., DNS-based request redirection. As stated in the [14] no standards are published yet for precisely identify the "nearest" location of service and manage DNS records according to the states of the servers. Nevertheless, companies such as Akamai, Netflix, and other big CDN companies do not publish their methods to which the security threats the companies are assuming. Therefore, dissertation consider the basics of the DNS-based redirection explained in [20] and [14].

As explained in Chapter 2 CDNs do not reveal their server selection criteria for their proprietary reasons. Alternatively, [14] states that there are three methods used by the CDNs to approximate the service location to their subscribers. Table 5.2 provides a summarized view of those three methods. According to limited available resources, as provided in Table 5.2, [14] pointed out that "connection monitoring" method is one of the well-known methods used by CDNs to approximate the content servers. Thus, the connection monitoring method approximates the TTL values of the DNS records according to the RTT time between the service location and probable neighbour, who might near to a connection initiation point, i.e., user. However, such methods do not consider the states of the servers, as well as the state of the underneath network to solve the "service point selection" problem effectively. As a matter of fact, RTT does not imply correct proximity of the users. Apparently, such methods might misjudge the location of the users. Moreover, as given in Table 5.3, CDNs use very short TTL values to provide fine-grain server selection. Thereby, CDNs always required their users to contact Auth. DNS servers and find the best position of service. However, such methods are unable to solve the service point selection problem adequately.

When considering the basics of CDN, both [14] and [20] state that CDN management servers, i.e., forward proxy server and Auth. DNS servers, determine the conditions of the content servers by (periodically) accessing the server states information of the content servers. The management servers then create redirection policies and recommend content servers to their subscribers by using DNS resolution [82]. Similarly, dissertation also assumed the same approach in the proposed method, and hence, as illustrated in Fig.5.3, the dissertation is proposed to advertise the server state information to the corresponding gateway router by updating the management servers or forward proxy servers. Moreover, the content servers are also programmed to advertise the server state information periodically to the attached gateway router using the server router communication protocol (SRC).

5.2.2.1 Server Router Communication protocol

The Server-Router Communication (SRC) protocol is an extension proposed to the ESLR. The SRC is deliberately developed to gather server state information and populate the server state information among the SoRs in the CDN. As illustrated in Fig.5.5, servers are particularly programmed to advertise the server state information periodically to their corresponding gateway routers using the header structure given in Fig.5.6. Since the protocol is designed as an extension of the ESLR, the ESLR uses its advertisement header structure presented in Fig.4.10 to carry SRC messages. Note that the ESLR adds the SRC messages to the ESLR header using the same method explained in [96].

As stated in Chapter 4, dissertation considers that it is the responsibility of the network administrators to configure the corresponding gateway routers of each content server. Once the administrators configure the gateway router address to a content server, the content server assumes a slightly altered method of "neighbor discovery process" of ESLR, which is explained in Chapter 4 Subsection 3.1,

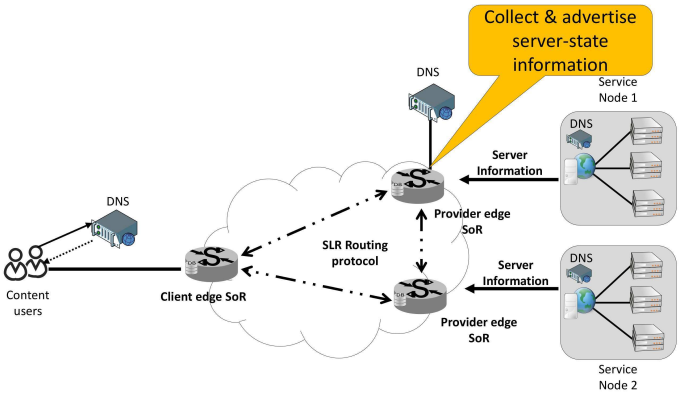


FIGURE 5.5: Provider edge routers collect and advertise server state information

```
-------------SERVER-ROUTER COMMUNICATION HEADER---------------
|      0        |      1        |      2        |      3      |
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---------------+---------------+---------------+-------------+
|           SEQ. NO             |N|S|SF|                      |
+---------------+---------------+---------------+-------------+
|                              MUE                            |
+---------------+---------------+---------------+-------------+
|                            LAMBDA                           |
+---------------+---------------+---------------+-------------+
|                         SERVER ADDRESS                      |
+---------------+---------------+---------------+-------------+
|                            NETMASK                          |
+---------------+---------------+---------------+-------------+
~                  SERVER CONTENT RECORD DETAILS              ~
~                                                             ~
+---------------+---------------+---------------+-------------+
```

FIGURE 5.6: Server-Router Communication header

to exchange the necessary protocol messages to initiate the communication link between the server and the gateway router (see Fig.4.6). For an example, initially server sends a `hello` messages to the the gateway router by setting the `Identifier` feild to `server` and the `Command` field to `SRC` (see Fig.4.4). Then the gateway router sends a `hello` message upon receiving the hello message from the server as a reply. When the server receives the `hello` message form the the gateway router the server sends another `hello` message by setting both `Auth_Type` and `Auth_Command` accordingly. Finally, the gateway router reply with same `Auth_Type` and `Auth_Command` and finishes the protocol negotiation process. Finally, servers periodically advertise the server state information to the gateway router using the header given in Fig.4.10.

The server advertisement message consists IP address, supporting URL(s) informational, average service rate (μ), average packet arrival rate (λ) of a content server, and a sequence number. The "server content record details," presented in Fig.5.6, is a free format, and that can be developed according to the requirement of the application. Consequently, this study assumes the URL/URI as

$< Protocol >:: // < SiteID >:< ServerID >:< TypeID >:< ContenID >$. When an SoR receives an SRC message from their intended CDN servers, the SoR calculates the average packet waiting time in the server buffer (($T_{S_n}(t)$) using Eq.(5.1), where μ_{S_n} is the average packet arrival rate, λ_{S_n} is the average packet service rate, and t is the time the advertisement packet generated. The assumption made for this calculation is the servers can be observed using M/M/1 queue model.

$$T_{S_n}(t) = 1/(\mu_{S_n}(t) - \lambda_{S_n}(t)) \tag{5.1}$$

When an SoR, i.e., gateway router, calculates the average packet waiting time at a server, it stores the corresponding information in its "server-state information" database. Figure 5.4 displays an instance of the table used to store the server state data. Note that; the database is created as an in-memory database. The table mainly has a record of each server maintained by the CDN provider. The proposed method makes sure that all gateway SoRs maintain up-to-date server state information table by advertising the server state information among the SoRs with the aid of ESLR. SoRs uses the ESLR neighbor discovery protocol to identify the neighbor SoRs. However, since the CDN networks are aware of their network infrastructures and the CDNs manages their network infrastructure, the study assumes that it is the responsibility of the network administrators to configure the SoRs with their neighbor SoRs. The neighbor SoR could be in several hop distance. However, the SoR uses unicast ESLR updates the configured SoR about the server state information. Note that SoRs use both periodic and triggered update advertisements to ensure quick distribution of the server state information among the other SoRs in a CDN. If an SoR does not receive update information about a particular server record, for the time of three periodic update intervals, corresponding SoR removes the server record from the server-state information database and notify the neighbors about the connection loss. Note that, if the ESLR configurable coefficient parameter "K1" is enabled on the SoR for the network where the servers belong to, the SoR considers the server record as a routing record and adds it to the routing table as well.

Fundamentally, as displayed in Fig.5.7, SoRs use the collected information to achieve three different objectives to solve the content navigation problem:

1. Recommend IP addresses of the least-busy servers by intercepting the recursive hierarchical DNS resolution process.

2. SoRs use the collected information for content-aware packet redirection. In this process, SoRs analyze the application layer information, i.e., URL and the content ID, of data request packets, and select the least-busy server from the server state information database; then, the SoRs forward the data packet to the selected server.

3. Use the ESLR to calculate the network paths according to network device states, i.e., delay, and thereby, deliver content using the minimal delay paths.

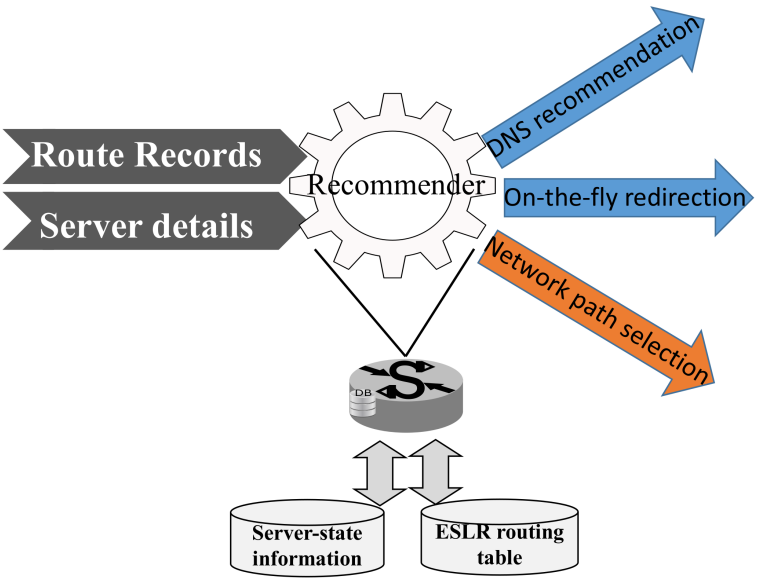


FIGURE 5.7: SoRs use collected information to enhance content navigation

SoR achieves the third objective by using the ESLR because the ESLR is developed to achieve the minimal delay network path selection. Since SoRs are collecting the server state information to maintain the server state table using the ESLR, SoR assumes that the $K1$ CCV of ESLR is always set to 1 and the routing table always has a record corresponding a particular content server. This approach enables the other routers in the network to select the minimal delay network path to a given content server.

5.3 Use collected information to leverage the DNS-based service point selection

As stated previously, most of the CDNs use DNS to find the service locations [18, 20, 21]. A detailed explanation about the DNS-based redirection is given in [20]. This study also assumes the same approach explained in [20]. Originally, in DNS systems, TTL values were defined to guarantee the cache time of the DNS record in the DNS servers. In general, TTL values are in the order of minutes or hours [53, 59]. However, with the introduction of DNS-based redirection into CDNs, the TTL values are set to smaller increments to aid CDN load-balancing and traffic flow optimization [53]. Such a phenomenon forces the large number of DNS query requests to be transmitted to the public network instead of served by the local DNS. When a given DNS strictly adheres to the calculated TTL, the local DNS servers must frequently traverse the hierarchical DNS servers to inform the subscribers about the change in the servers [20, 53, 59].

Generally, in the DNS-based redirection, the local DNS server recursively queries a hierarchy of DNS servers until it reaches the Auth. DNS server for the CDN domain [14, 20, 53, 59]. The caveat is the time that has to be spent in finding the Auth. DNS server profoundly affects the connection initiation

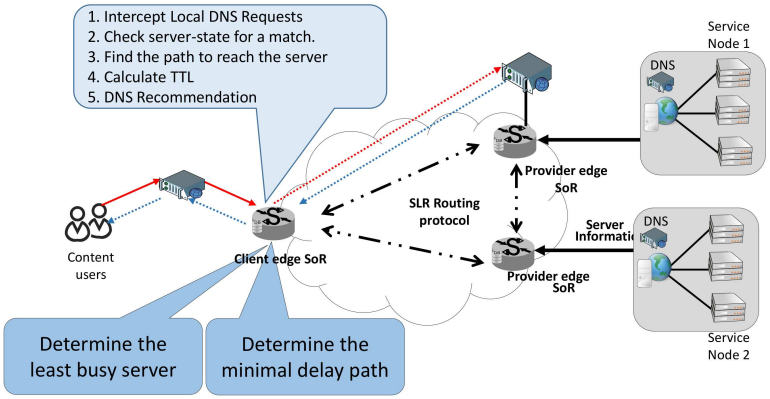


FIGURE 5.8: Leverage DNS-based redirection

and rapid adaptation to the changes occurring in servers. Moreover, the DNS based redirection assumes that the local DNS server is always staying with the client and, most of the times, the Auth. DNS servers select the best service location based on the statistics they measured using connection-monitoring methods. Table 5.4 displays a simple survey the author conducted to understand the behavior of DNS redirection. The table shows the measured distance (from the client), time to reach the DNS server, and the TTL values those Auth. DNS servers provided for selected popular CDN applications.

Let us assume the Keio university's DNS server is placed within the campus network where the local computer is also placed on. All other DNS servers are at least one hop away from the local computer when comparing DNS server placed in local network. Moreover, the TTL values are always below 100 s, and that means the customers should contact the DNS servers within every 100 s if they wanted to get the latest server information. Such kind of recursive, resolution process pointed out the growth of the DNS traffic in a public network, which is depicted in Fig.2.8. Another notable point is that both IEEE Explore and Facebook services use same CDN network but two different DNS servers. That recapitulates each service maintains their own Auth. DNS servers and the server selection problem is addressing based on the place where the Auth. DNS server is placed on. Consequently, as pointed out in [14], the location determination requires some standards.

5.3.1 Proposed algorithm to leverage the DNS-based service point selection

To address limitations mentioned in Section 5.3, as depicted in Fig.5.8, the edge router, i.e., user gateway router, is programmed to intercept DNS query messages that are transmitting to the ISP network from the local network. By intercepting the DNS queries, SoR recommends DNS records to the Local DNS server according to the busyness of the server and the adequate network path to reach that server. However, SoRs do not intercept DNS response messages and secure DNS query messages. Moreover, if the SoR is unable to resolve a DNS server query, the SoR passes it to the Auth. DNS server and assume the typical DNS resolution process. The reason that the SoRs are used only for leveraging

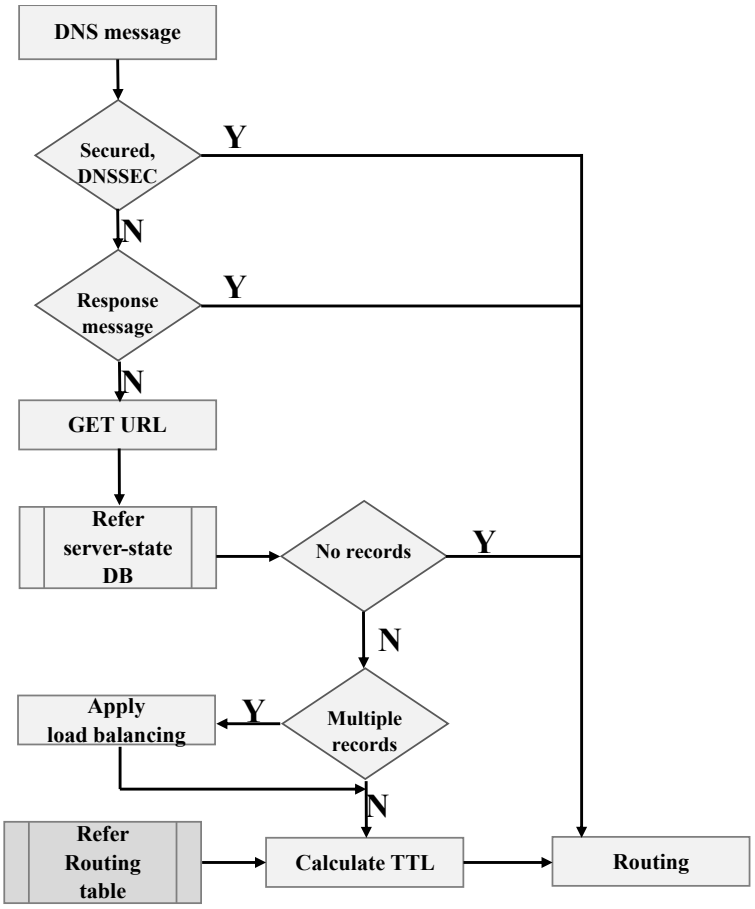


FIGURE 5.9: Proposed algorithm for DNS recommendation

TABLE 5.4: Analyzing the behavior of DNS in real network

Service	Server name	Distance(Hops)	TTL(s)	Delay-to-reach
Keio	ks3prmw3.lib.keio.ac.jp	7	3081	2ms
YouTube	ytimg.l.google.com	10	279	2.25ms
Facebook	scontent.xx.fbcdn.net	18	13	194ms
Akamai Auth.DNS	a72-246-188-35.deploy. akamaitechnologies.com	8	20 - 200 (Max)	3ms
IPL (VoD)	datacdn.iplt20.com	10	13	10ms
Gmail	googlemail.l.google.com	10	35	2.5ms
IEEEExplore	e1630.c.akamaiedge.net	8	15	2.5ms
Mozilla	tiles.r53-2.services.mozilla.com	18	27	130ms

the DNS-based redirection by avoiding the limitations of the regular DNS-based service location selection. Moreover, to recommend the DNS records based on both server states, and the network state information. Therefore, the proposed method can be applied to the existing CDNs without doing any additional modification, but placing an SoR at the edges of the networks. Thus, the proposed method is designed to co-exist with the generic CDN architectures.

The algorithm used for DNS recommendation is presented in Fig.5.9. As shown in the figure, when an SoR receives a DNS message, first it checks for the response or secure DNS messages. In either case, SoR forwards the message to the intended destinations without further analysis. For the DNS request messages, SoR extracts the URL/URI mentioned in the DNS question section. An assumption is made that the URL is in following format: $< Protocol >:: // < SiteID >:< ServerID >:< TypeID >:< ContenID >$. Next, SoR checks the server-state database by using the content ID to find the matching server records. If SoR finds more than one matches, the SoR apply the load balancing algorithm and finds the server that has the minimal packet waiting time in its buffer. After that, using Eq.(5.2), SoR calculates the TTL value of the DNS record accordingly. Note that, when calculating the corresponding TTL values, if the ESLR is used as the routing protocol, and $K1$ is enabled for the server record, the $(T_{S_n}(t))$ will be $(T_{S_n}(t)) + ep$ delay. Finally, SoR creates a DNS response message and reply to the local DNS server.

$$TTL = K \times t_S \times \left(\frac{1}{\rho_S}\right) \tag{5.2}$$

There is neither standard nor evidence can be found to calculate the TTL values apart from the guesstimate presented in Table 5.3. Therefore, this work proposes Eq.(5.2) to calculate TTL values, where $(\rho_S(t))$ is server utilization and $(T_{S_n}(t))$ is average waiting time. Subsequently, Eq.(5.2) uses both server utilization and the average packet waiting time to calculate the corresponding TTL values for content servers. Note that $(T_{S_n}(t))$ is calculated using Eq.(5.1). The coefficient K changes according to $(T_{S_n}(t))$ and is introduced to convert the TTL value to the order of seconds. Based on several trial-and-error methods, the dissertation uses $K = 1,000$ as for the experiments. The inverse of the server utilization $(\rho_S(t))$ is used in Eq.(5.2) to adjust the TTL value according to the server usage, i.e., small TTL values for busy servers and large values of TTL for idle servers. The assumption made is the servers can be represented using M/M/1 queue model. Using a simple derivation, we found that the optimal TTL value could be calculated with the packet arrival rate is two time as the packet service rate. Furthermore, Eq.(5.2) demonstrates a better ability to adapt to conditions (e.g., server changes) and to recommend content servers to the end-users adequately.

5.4 Use collected information for on-the-fly packet forwarding

As previously stated, one of the primary motivations of the SoR project is to provide content based services for the end users. As pointed out in [15, 42, 44], online gaming, video streaming, and audio streaming applications are the most popular CDN applications. Streaming media define the transport layer protocol according to the user requirement and quality. For an example, CDNs use Real time streaming protocol (RTSP) for most of the streaming applications [144] and maintain bi-directional

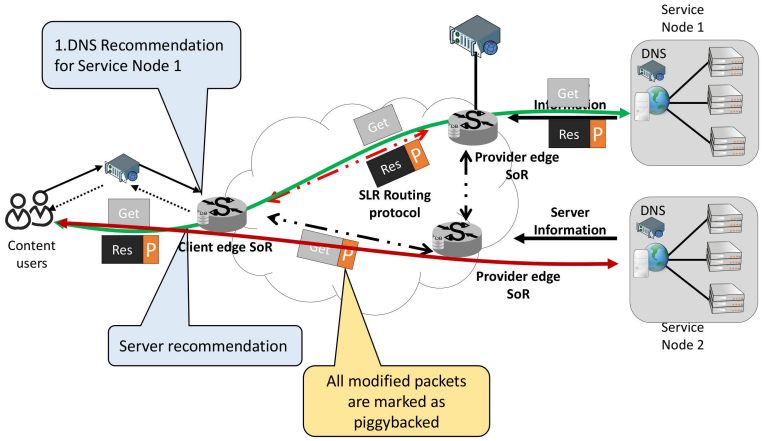


FIGURE 5.10: On-the-Fly packet forwarding

communication. Moreover, online gaming, one of the most popular applications use CDN for content navigation, also uses such sort of UDP based protocols and most of those protocols are request-response protocols [145]. The limitation is that, when the CDN strictly adheres to a DNS system, the DNS-based server selection method does have the capability to update the clients according to the server states unless the clients request a new service point upon expiration of the corresponding DNS record. This factor limits the service quality when users are redirected to the busy servers.

Hence, working as an intermediary, SoRs get the server state information periodically. Namely, Auth. DNS servers, and reverse proxy servers are configured to update the corresponding SoR with the connection monitoring data. Moreover, since the SoRs get the support of the routing protocol to advertise the server state information, such data can be transmitting across the CDN faster than the TTL values. So that the user gateway routers can intercept the data request packets and check whether the packets are addressed to the best available service point. In that case, as illustrated in Fig.5.12, if the application permits to alter the header information and redirect the packets in the middle of a transmission, the gateway router can redirect the data request packets to the server which has the minimal delay of service time through the minimal delay network path. Note that the dissertation uses Type of Service (ToS) bits in the IPv4 header to control on-the-fly packet forwarding capability via the client application.

This approach can be used the reduce one of the main limitations of DNS, which is redirecting the users without heavily depending on the TTL expiration time. The main limitation of the proposed approach is that it is profoundly contingent of the transport layer protocols. For an example, original TCP protocol does not allow such redirection in the middle of a connection [146]. However, although a CDN is a distributed infrastructure, the entire infrastructure belongs to a single owner, and he controls the whole infrastructure. Therefore, the owner has the capability of building the infrastructure satisfying the transport layer concerns. Moreover, companies such as Cisco and Hitachi identified the importance of enhancing on-the-fly TCP redirection, and they have provided commercial router solution for gateway routers such as session border controller and Hitachi WAN accelerator respectively [32, 33].

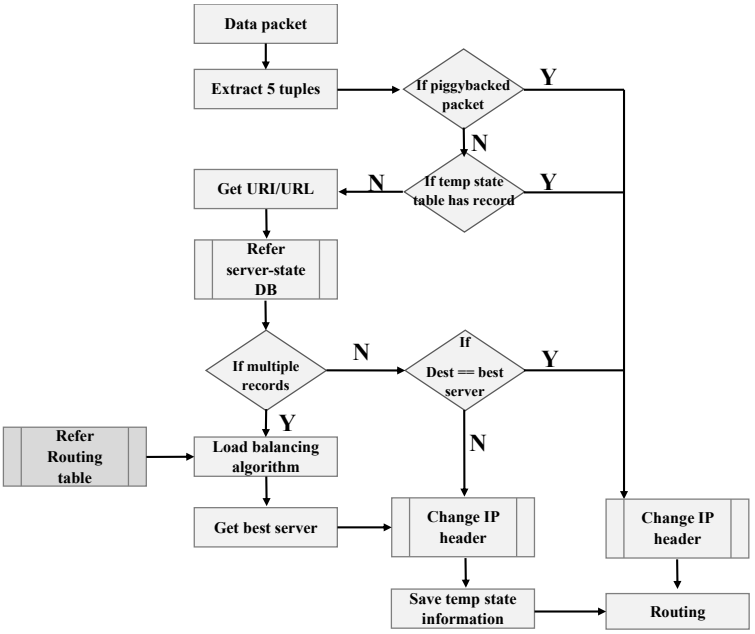


FIGURE 5.11: Proposed algorithm for on-the-Fly packet forwarding

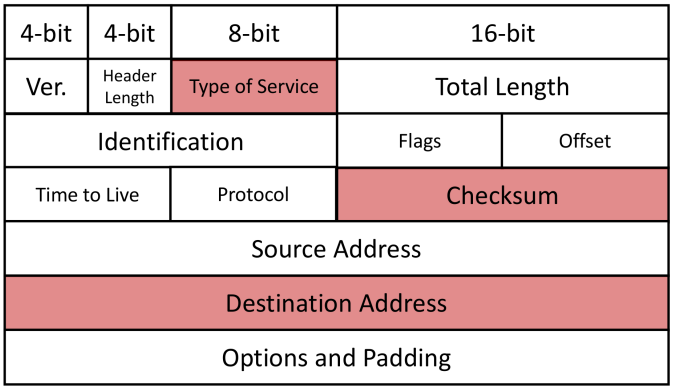


FIGURE 5.12: IPv4 header

TABLE 5.5: IPv4 ToS values and the defined services

Flag Value	Action
0x00	Do not permit to modify the destination
0x01	Permit to modify the destination
0x02	The destination has been modified on the way

5.4.1 Proposed algorithm for on-the-fly packet forwarding

The main scenario of on-the-fly packet redirection is presented in Fig.5.10. The algorithm used for altering the header structures according to the Application layer information is illustrated in Fig.5.11. To execute the proposed method, this study used the first three bits of Type of Service (ToS) field of IPv4 header to provide the permission to alter the IPv4 header in the middle of the transmission. This study uses the ToS field as presented in Table 5.5.

As shown in Fig.5.10, when a request packet arrives at the client's gateway SoR, the SoR analyze the packet for its application layer information. SoR checks whether the packet is destined to the best available servers. If not, the gateway router redirects the packet to the best available server. The altered packets are marked using the piggyback technology, i.e., set ToS to 0x01, to let other SoRs know that this packet has been altered in the middle of the transmission. Furthermore, the reply packets are also modified using the same ToS value to skip packet analysis process. The transport layer was assumed as user datagram protocol (UDP) to avoid the protocol complexity. The Same scenario can be applied to achieve on-the-fly TCP redirection when the SoRs uses techniques described by Hitachi WAN accelerator [63], Cisco session border controller [32], and man-in-the-middle TCP redirection [147]. However, due to the protocol complexity, this study does not assume dynamic redirection of TCP streams.

The algorithm used for content-based dynamic packet redirection is presented in Fig.5.11. According to the algorithm, when a data packet comes to the SoR, the SoR first analyses the packet for five tuples. After that, SoR checks for the piggybacked packets and ignores them. If the packet is not piggybacked, the SoR then checks for the existing record in its temporary redirection state table. If a record is found, which matches the content ID that the user is requesting, SoR alters the packet's destination address using the address found in the table. After that SoR sets the ToS and the Checksum of IPv4 header and route the packet to the new destination. If no record is available in the temporary redirection state table, the SoR then checks for the server-state database using the content ID, gets the available server state records. If only a single record is found, SoR then checks the packet's destination address is same as the obtained address. If the destination address is same, SoR routes the packet. Otherwise, SoR uses load balancing algorithm and check with the ESLR table to get the server, which has the lowest packet waiting time and the minimal delay path to reach the server. Then SoR changes the IPv4 header's destination address, ToS, and the checksum values as depicted in Fig.5.12 (all modified sections of the IPv4 header is highlighted in "red" color in Fig.5.12). Finally, SoR creates a temporary record about the modification on the temporary redirection state table, and forwards the packet to the next hop towards its new destination. Note that the data packets contain an application layer Protocol Data Unit (PDU), which has the intended URI. For an example, URI can be arranged as: $< Protocol >:: // < ServerID >:< PORT >:< ContenID >$.

5.5 Control plane message flow of the collaborative infrastructure

This section describes the message flow among the client, DNS servers, intermediary SoR, and the CDN network. The basic diagram is illustrated in Fig.5.13. There are three scenarios when the SoR works as an intermediary to leverage the DNS-based service point selection and network path selection. 1) Local DNS have a tentative location of a server which matches the client's request, 2) Local DNS does not have the location that matches the client's request, and 3) Neither Local DNS nor client's

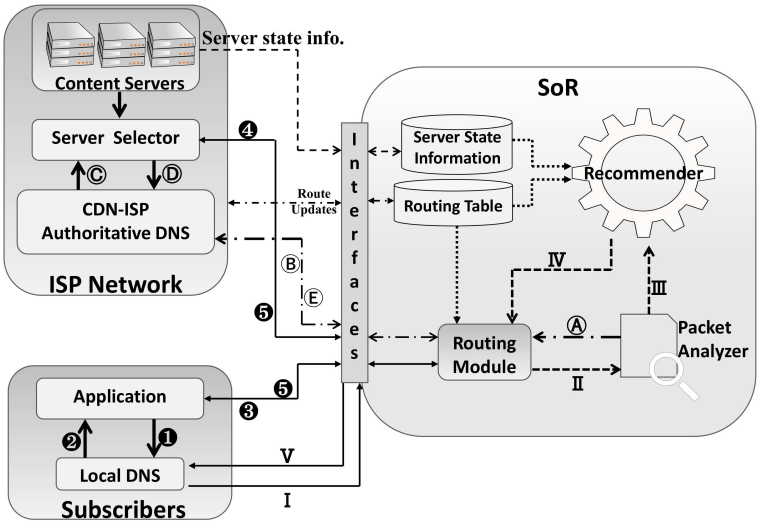


FIGURE 5.13: Control plane message flow among local network, gateway router, and the CDN

gateway SoR do not have the location information. Following subsections provide the message flow involved in above mentioned three scenarios and how SoR and DNS-based service point selection methods collaboratively navigate content according to states of both servers and the network. Note that; the dissertation assumes the CDN infrastructure allows the dynamic content-based redirection by setting the ToS field of the IPv4 header as explained in Section 5.2.3.1 and presented in Fig.5.12. The following explanations assumes the message flow presented in Fig.5.13.

5.5.1 When Local DNS possesses a service point

❶ User contacts the local DNS server for DNS resolution. ❷ Since local DNS server has tentative service point, regardless of the distance and the server states, the local DNS replies the IP address to the user. ❸User sends the data request packets to the selected destination by setting the ToS as 0x01 after receiving the IP address from the local DNS. When the packet reaches the gateway SoR, at this point, the edge SoR intercepts the data request packets. Both packet Analyzer and the Recommender module execute the algorithm presented in Fig.5.9 and follows up the necessary steps: II**.**, III**.**, IV**.**, or (A) accordingly. If the packet type is not supported or the packet is not enabled for content based redirection, Analyzer module passes the packet to the routing module and assumes the existing CDN strategies. Otherwise, the Recommender module checks whether the packet is addressed to the best server available in the server-state database and modify the IPv4 header accordingly. ❹ and ❺ SoR finally forwards the packet to the destination. Note that the reply packets are always followed step (A) because those are not intended for further analysis and modifications.

5.5.2 When local DNS does not possess a service point

❶ User contacts the local DNS server for DNS resolution. **I.** Since the local DNS server does not possess a service point matches the client's request, the local DNS server sends a DNS query message to the CDN's Auth. DNS server through the ISP's DNS server. Note that the request message might be addressed to Root or TLD servers. However, for the simplicity of explanation, only ISP DNS server is assumed. When the DNS request packet reaches the client's gateway SoR, at this point, the gateway SoR intercepts the DNS query message. SoR then uses both Analyzer module and the Recommender module to execute the algorithm presented in Fig.5.11 and executes the necessary steps: **II.,** **III.,** IV**.,** or (A) accordingly. If the DNS packet uses security protocols, the Analyzer module assumes step (A) and SoR assumes the regular hierarchical DNS resolution process. Otherwise, the Recommender module creates a DNS response message by using the necessary DNS header structures. Note that the TTL value is calculated according to the explanation given in Section 5.3.1. V**.** The SoR forwards the DNS response message to the local DNS servers. ❷ When Local DNS server receives the response message, the Local DNS sends it to the end-user. ❸, ❹ and ❺ Finally, the user downloads the content.

5.5.3 When neither local DNS nor SoR does not possess a service point

❶ User contacts the local DNS server for DNS resolution. **I.** Since the local DNS server does not possess a service point matches the client's request, the local DNS server sends a DNS query message to the CDN Auth. DNS server of the ISP. Note that the request message might be addressed to Root or TLD servers. However, for the simplicity of explanation, only ISP DNS server is assumed. When the DNS request packet reaches the client's gateway SoR, at this point, the gateway SoR intercepts the DNS query message. SoR then uses both Analyzer module and the Recommender module to execute the algorithm presented in Fig.5.11 and execute the necessary steps: **II.,** **III.,** IV**.**, or (A) accordingly. When the Recommender module determines that the SoR does not have a service point, using steps ((A) and (B)), SoR forwards the DNS query to the CDN Auth. DNS servers in the ISP. ((E) and V**.**) The Auth. DNS server resolves the IP address and replies to the local DNS. ❷ When Local DNS server receives the response message, the Local DNS sends it to the end-user. ❸, ❹ and ❺ Finally, the user downloads the content.

5.6 Simulations, test results, and discussion

This section provides the simulation implementations, yielded results and discusses the results to show how the collaborative architecture enhances the content navigation. In particular, the results were obtained to examine user performance, server utilization improvements, and network resource utilization. The performance of CDNs is heavily contingent of the CDN infrastructure and the ISP topology. However, neither ISP topologies nor CDN infrastructures publish their topologies to which the security threats both CDN and ISP are assuming. So, implementing simulations to evaluate CDN architectures, which are similar to big CDNs such as Akamai, are impossible due to the unavailability of the Intel.

Alternatively, there are several options available to implement simulation topologies using freely available educational backbone networks such as WIDE Project's network topology [148], router-level ISP topologies such as Rocket Fuel [133], and virtualization topologies such as JGNx [149] and Planet Lab [150]. Even though publically available educational networks provide their network topologies for free of charge, due to the operational and legal constraints, those networks do not permit to place computers on their network for testing purposes. Nevertheless, as SoR is still under research and development, it is practically impossible to place SoRs in such commercial networks. Therefore, the only available option is to implement simulation scenarios using realistic network simulators such as ns-3 using the topology data and available link details. The thesis study implemented two simulators to evaluate the proposed method of using SoRs to make collaboration between ISP and CDN, and thus, use collaborative infrastructure to address the content navigation problem. A summary about the two simulators are given in Table 5.6.

The first simulator was implemented using Planet Lab network. The reason why this study uses Planet Lab is that the Planet Lab network offers a virtual slice of computers to perform experiments. Consequently, the virtual slice can be used to implement the data plane network of CDN by implementing clients, servers, and the SoRs. However, the Planet Lab network manages the routing, and the virtual slice has no control over the routing plane. Therefore, Planet Lab network provided the best simulation environment to demonstrate the uncooperative environment between the CDNs and ISP because the Planet Lab network controls routing and the CDN infrastructure is controlled by the author, which simulates the most common CDN approaches. However, in the ns-3, since both CDN and ISP can be configured by ourselves, the total system can be implemented as a one. Therefore, thesis study implemented a CDN network on ns-3 using WIDE network's topology. The simulation topology is implemented to simulate a full collaborative CDN environment such as Telco CDNs. Both ISP network and the CDN was implemented by using the same topology. Thus, routing plane and CDN's data plane are controlled by one system. Therefore, this study conducted an extensive simulation using the ns-3 simulator and evaluated the proposed infrastructure.

There are several assumptions have been made when implementing both simulators.

TABLE 5.6: Summery of the implemented simulators

	Planet Lab	ss-3 (using WIDE network topology)
ISP	Considered as the planet lab network infrastructure.	ISP network is implemented using the WIDE network topology and its configuration details
ISP network	Due to the limited resource availability, entire Asia pacific as a one ISP	Since the WIDE deploys their routers all over the Japan, entire japan as a one ISP
Implementation	CDN network is implemented using a virtualized slice of Planet Lab	The CDN network is implemented as a part of the ISP network
Routing	Controlled by Planet Lab	ESLR is implemented to the simulator
SoR	Placed according to the server and clients placement	Placed according to the server and clients placement
Content server placement	Placed two surrogate servers for each country	Placed surrogate servers according to the popularity of the city
Users placement	Placed users assuming whole country (used only clients placed at Japan)	Placed users randomly
Network links	Have no control	Implemented according to the WIDE network topology

TABLE 5.7: Parameters used for Planet Lab simulation

Parameter	Value
Number of servers	5
Number of traffic generators	10 (in Japan)
Packet generation rates	100 pps - 500 pps
Servers advertisement interval	120s
DNS resolution time	20s - 200s
Bandwidth of the links	As presented in [151]
Assumed number of content	1000
Average packet size	512B

1. Only one ISP network is simulated in both scenarios.
2. All servers in the CDN network are fully replicated from the main server.
3. All servers and users are connected to the ISP network using SoRs as their gateway routers.
4. If ISP uses ESLR as the routing protocol, $K1$ ESLR configurable coefficient value is configured on the necessary gateway routers to create a server record in the ESLRs routing table.
5. CDNs maintain the DNS architecture as explained in FreeFlow DNS [59].

5.6.1 Simulation implemented using Planet Lab

Figure 5.14 displays the simulation topology implemented on the Planet Lab network. As illustrated in the figure, this study assumed that entire Asia Pacific region as a one CDN, and the content servers was

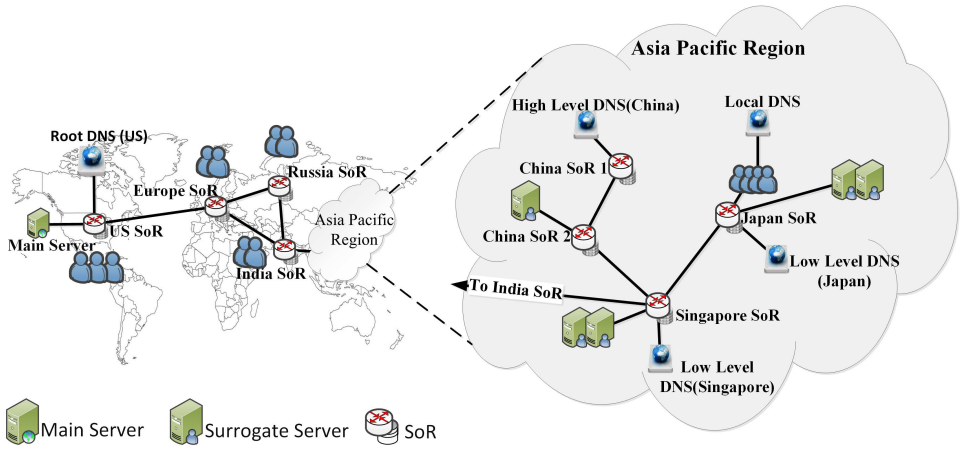


FIGURE 5.14: Topology implemented on the Planet Lab [150]

placed representing each country. The content servers were placed in Singapore, China, and Japan. DNS servers were placed within each country to represent the Auth. DNS, and the study assumed that the DNS hierarchy follows the Free Flow DNS [59]. Note that the main server was placed in the US. Clients were placed along side with local DNS servers, and the client was programmed to contact the local DNS server before fetching content. Note that a simple packet structure is implemented to emulate the DNS packets.

As illustrated in Fig.5.14, both content servers were connected to the CDN using SoR as their gateway routers. The servers were implementing by assuming streaming servers and the servers were programmed to send data packets when the client's requests reach the content servers. Servers were programmed to advertise the content server details to the corresponding gateway router. However, since the planet lab network does not provide an option to implement a routing protocol, i.e., ESLR, the study implemented a simple UDP socket communication protocol to advertise the server state details periodically. Further, SoRs also used the same socket communication protocol to distribute the server-state details among the SoRs. SoRs were configured with a static route records and assumed that as the routing plane of ISP.

As Fig.5.14 displays, only the clients placed in Japan considered evaluating the proposed architecture. Clients were implemented as random rate packet generators by assuming a streaming service as the application. In essence, both clients and servers were implemented as a request-response data communication application. Clients were configured to use an URI as follow: $rtsp$: // < $SurrogateServerAddess$ >:< $Portnumber$ > / < $contentID$ >. Moreover, Clients were programmed to change the content id randomly during the data transmission. In each change, clients were programmed to contact the local DNS server to resolve the DNS. The simulation assumed that the clients used content ID to resolve the DNS. The data packets size, both request, and response, were assumed as 512B.

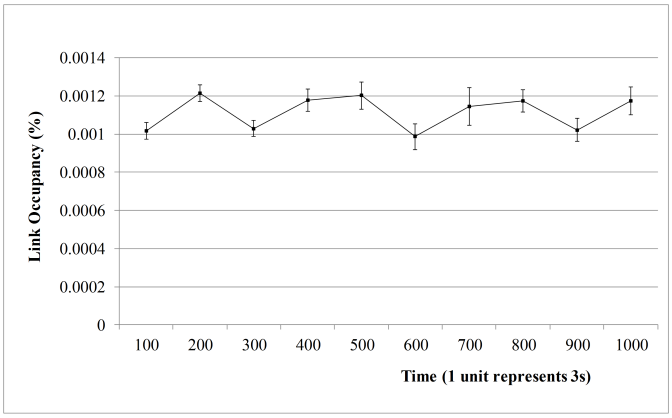


FIGURE 5.15: Link usage for control messages

Two simulation scenarios were conducted using this simulator. Parameters used for the simulations are given in Table 5.7. The first experiment assumed the existing CDN architectures, where the CDNs use a hierarchy of DNS servers to find the content servers and redirect users to the selected content servers. Similar to the DNS resolution, the Auth. DNS servers were configured to issue the server address using the round robin algorithm. Moreover, clients were programmed to contact local DNS server at each change of the content Id, and the local DNS servers are programmed to contact hierarchy of DNS servers to get to the Auth. DNS servers. The second simulation was conducted by assuming the collaborative approach. SoRs intercept both DNS packet and user data request packets perform DNS recommendation and content-based data packet redirection. Similar to the first simulation scenario, clients were programmed to contact local DNS servers when they are changing the content ID, and local DNS servers were programmed to resolve the DNS iteratively. The proposed method is named as DNS SoR collaborative Redirection (DSCR).

5.6.1.1 Experiment results

Link usage for control messages The first experiment was conducted to understand the resource usage of advertising server state information among the SoR within the CDN. The simulation was executed about 3,600 s. As stated previously, Planet Lab maintains its network routing structure, and it does not allow controlling the routing environment. So CDN servers and the SoRs maintain its simple UDP-based communication protocol to advertise the server state information. Consequently, some protocol packets passed through each interface, and the average protocol message sizes were calculated for a given time duration; measured in 100 s time buckets. After that, the bit rate was calculated using Eq.(5.3). Finally, the found bit rate was divided by the link bandwidth using Eq.(5.4) and plotted on the graph presented in Fig.5.15. As explained in [151], Planet Lab guarantees about 6Mbps to 10Mbps for each virtual slice. Thus, this experiment assumed that the network bandwidth is as 5 Mbps.

$$bitrate = (numberofcontrolmessages \times Avg.Packetsize)/duration \tag{5.3}$$

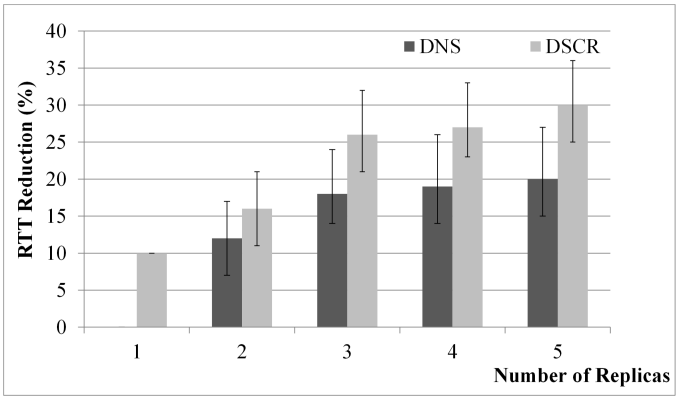


FIGURE 5.16: Percentage of RTT reduction

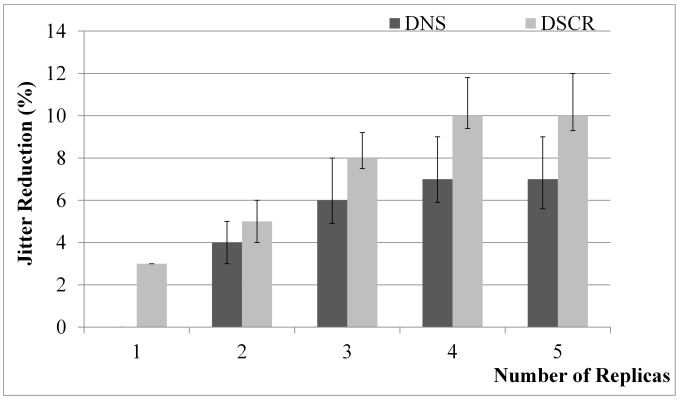


FIGURE 5.17: Percentage of jitter reduction

$$linkusage = bitrage/linkbandwidth) \times 100\% \tag{5.4}$$

As per the experiment, only three packets were used by the communication protocol to advertise the server state information about a server in each advertisement interval. The reason can be explained as follows: assume a content record is 4Bytes, and the link MTU is 1500B. In that case, 360 records can be added it to one advertisement packet. Even if we assume that the content server has all 1,000 content in its storage, it uses only three packets to advertise about its server record state information. Therefore, as presented in Fig.5.15, average bandwidth usage for the protocol management is about 0.001% of the network link. This is considerably small because the route advertisement messages are not considered for this calculation. The reason is that the Plant Lab network totally manages the route advertisements. However, given that the network links are in gigabit scale, and links are continuously updated regarding bandwidth the link usage is in an acceptable range, even for the large scale CDNs, the link usage can be maintained under an acceptable range. The main reason is that the SoRs are proposed only to advertise the server state information, not the content.

User experience The simulations were assumed that the client and servers are designed to perform streaming services. The quality of the streaming services depend on the packet arrival time, i.e., round

trip time (RTT), and the packet arrival jitter. Therefore, those two parameters were used to measure the user experience quantitatively. To measure the RTT, five experiments were carried out by increasing one server at a time. In each experiment, five simulations were performed and measured the average RTT and plotted the obtained results in Fig.5.16. According to the figure, RTT was reduced when the number of servers in the network was increased. The reason is that both methods have redirected the users to the available servers and effectively reduced the content download time. However, when considering the collaborative approach, the reduction percentage is high compared to the traditional DNS-based user redirection. The reason is that the SoRs effectively reduces the time to resolve the DNS resolution time. Moreover, since SoR determine the minimal busy server, the users have the opportunity to download the data from the least busy server. Consequently, when the CDN has five content servers, the RTT was reduced by 10% compared to the traditional DNS-based approach. Moreover, the rate of reduction of the RTT with respect the number of servers is also high in the collaborative method compared to the DNS-based method.

Same simulation scenario is carried out to measure the jitter reduction as well. The obtained results are plotted in Fig.5.16. Similar to the RTT, the jitter also reduced when increasing the number of servers in the CDN. Again, the reduction rate is about 4% high in the collaborative approach. The reason is that the SoRs redirected the user to the least busy server by intercepting both DNS and content request messages. Therefore, the effective jitter is also reduced in the proposed method. However, jitter reduction is comparatively low because the implemented topology does not have control over network path selection. Hence, the main source that can be used to reduce the jitter is network path selection.

According to the results presented in Figs. 5.16 and 5.17, it can be concluded that if CDN infrastructures use edge routers to leverage the DNS-based server selection according to the server state information, the effective user experience can be increased. Furthermore, if the TTL values of the DNS messages are calculated according to the server states, the users can be redirected to the best server regardless the proximity. This improves the user's performance up to some extent even the CDNs are not totally collaborated with the ISPs. In summary, as pointed out in [26, 49, 73], the collaborative approach of server selection improves the performance of the users effectively.

Server utilization The server utilization was measured as the next experiment. The server utilization was measured by measuring the servers queue sizes by incrementing a number of servers. Five experiments were conducted, and the average results are considered to plot the graph presented in Fig.5.18. As shown in the figure, when the number of servers was increased in the CDN, both collaborative method and the DNS method displays a reduction in queue sizes of the servers. It is notable that after turning on three servers, the server queue size is almost stabilized in the DNS. The reason is that the DNS was configured to offer the servers based on RR algorithm, which is the basic algorithm used in the DNS. Moreover, DNS have to wait until the users contact the Auth. DNS servers upon the

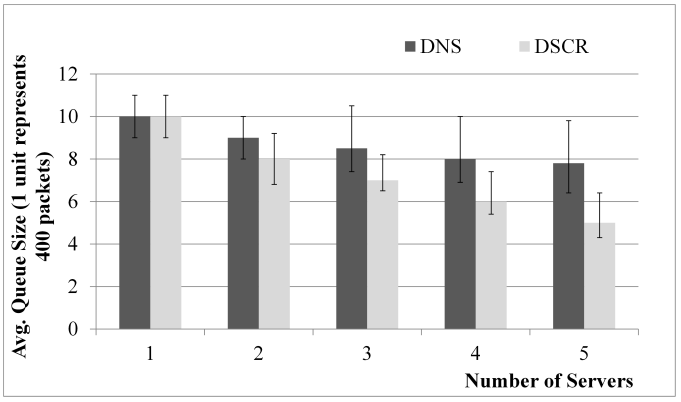


FIGURE 5.18: Server utilization

TTL expiration. However, when considering proposed method, as the figure displays, the collaborative approach was able to reduce the number of packets in each server when the number of servers was increased. That is because the server-router communication protocol was able to update the SoRs in the network about the new servers within small time scales. Therefore, user edge gateway routers are always up to date about the state of the servers and redirects the users to the least busy server using both content-based redirection and leveraging the DNS-based redirection. This phenomenon effectively reduces the queue length of the servers. Note that; this process can be assumed as SoRs run the least load algorithm to enhance the user redirection.

In summary, the results presented in above sections summarize the proposed approach, using edge routers to determine the least busy servers and redirect the user to the selected servers, can effectively provide benefits for both users and servers. Moreover, the results show that the SoRs can work as a successful intermediary to leverage the DNS-based user redirection, and SoR can be used within a network without incurring any changes to the protocols or the applications of the CDNs. The only consequence is the SoRs or some routers similar to SoR have to be placed as the gateway routers of the CDN. If we assume such networks, the proposed method can be used to enhance the content navigation process with providing benefits to both users and the servers. However, the benefits are coming at the cost of link usage to exchange the control messages used by both servers and routers to advertise their state information. Given that the protocol messages use an acceptable bandwidth compared to the modern network, the benefits of introducing the gateway routers to improve the content navigation is high.

However, the network path selection cannot be achieved due to the configuration strains in the Planet Lab network. Thereby it limited the evaluation of the proposed architecture because, when considering the content navigation problem, it is the harmony of network path selection and the service point selection. It is understandable that the use SoR as a gateway router gives certain benefits for service point selection, and hence, makes considerable improvement for both users and servers. However, it is inadequate to conclude the total advantages of the proposed architecture without considering the network path selection process. To this end, the thesis study implemented a CDN on the ns-3 simulator

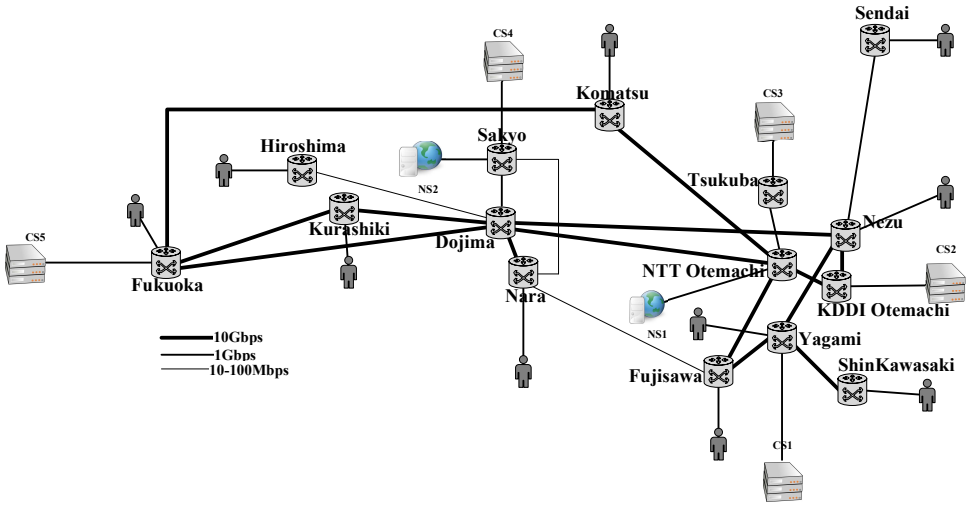


FIGURE 5.19: Topology implemented on ns-3 using WIDE network topology [148]

using the network topology of WIDE network. Since the Telcos are hosting their CDNs now, the WIDE network and the proposing CDN network better match the modern CDNs. The thesis study then used the implemented CDN network on the ns-3 simulator to perform a substantial quantitative evaluation of the proposed collaborative approach.

5.6.2 Simulation implemented using ns-3

Exposing internal router-level topological information as well as the server-level infrastructural information remains unsafe due to the vulnerabilities to which both the ISP and CDN are facing. As a consequence, ISPs make topological data are publically unavailable. Consequently, this study uses an education backbone network presented in [148] to implement a topology illustrated in Fig.5.19. Though the topology is not an ISP network, to study the behavior of the SoR as a gateway router, set of realistic simulations were arranged by configuring the same network conditions (e.g., link speeds) as provided in [148]. Moreover, since the CDNs are an overlay network built on top of a packet network, it is not necessary to have an ISP network to implement a CDN. Thus, simulating the overlay network is enough to study the behavior of the proposed collaborative architecture. Several simulations were carried out to evaluate the proposed system regarding resource usage for control messages, user behavior, network resource utilization, server utilization, the behavior of the DNS architecture, and the adaptation to the server changes. The parameters used for the simulations are presented in Table 5.8. For the simplicity of presentation, we assumed that one CDN operates within an ISP network.

Similar the planet lab network simulation, 1,000 items of content were assumed, and the contents were fully replicated among the content servers. The content servers were placed on the CDN based on the popularity of the cities by assuming a number of content users are high based on the popularity. Since the CDN is implemented as a Telco network, the hierarchy of DNS servers were also assumed to the Free Flow DNS. Particularly, two Auth. DNS servers were used for the simulation network, and they

TABLE 5.8: Parameters used for Planet ns-3 simulation

Parameter	Value
Number of servers	5
Number of traffic generators	10
Traffic generation rates	–
Servers advertisement interval	120s
DNS resolution time	20s - 200s (As per Table 5.4)
Bandwidth of the links	As presented in [148]
Assumed number of content	1000
Average packet size	256B

placed in Otemachi and Sakyo. Altogether 13 SoRs were used as the gateway SoRs of both end-users and content servers. The end-users were created as random traffic generators and were randomly placed on the implemented network. The end-users were programmed to resolve the DNS when the TTL values expired. They generated data request packets based on the assumption of a video streaming service, which is a well-known application in CDNs [42, 44]. To maintain backward compatibility with the existing Internet architecture, we assumed that content is requested by using a URI. Subsequently, the users generate UDP-based data request messages as explained in [152], and the data request message carries the intended URI, i.e., $rtsp://<SurrogateServer\_Address>:<Portnumber>/<contentID>$, that the end-user wanted to fetch.

The network topology was implemented according to the topology information given in [148]. However, [148] does not provide link delay information. Therefore, the topology is configured by assuming the same delay for each link. ESLR is configured on the network. To generate burst traffic conditions, the client programs were configured to change their traffic generation rates randomly and turn on and off the clients randomly. All ESLR CCVs were set to 1 and the network path selection considered network links, routers, and the servers for traffic metric calculation. Real DNS implementation is assumed for these experiments, and the DNS is implemented to the ns-3 simulator according to RCF 1035 [56].

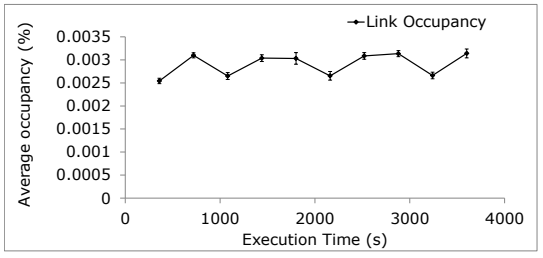
5.6.2.1 Experiment results

According to Table 5.8, the network consists of thirteen SoRs and two regular routers, i.e., ns-3 nodes, and ESLR was configured as the IGP of the topology. As a consequence, both SoRs and regular routers worked in harmony and used ESLR advertisement messages to calculate the route metric without any malfunction. Moreover, the server router communication protocol was used to advertise the serer-state information among the corresponding gateway routers. The Server-router communication messages were spreader among the edge routers using the same ESLR advisement header structure and using the same communication channel, i.e., same UDP socket.

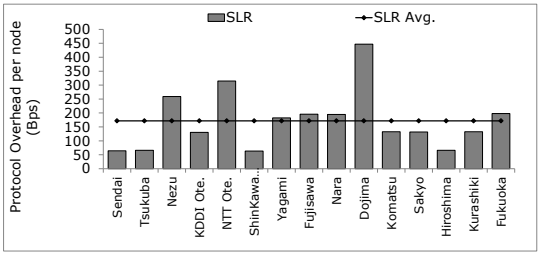
Similar to the first experiment, the topology was simulated about 4,000 s period and was evaluated for the effectiveness of using ESLR for network path selection. Note that; some links were disconnected

during the simulation time to measure the effectiveness of the ESLR for network path selection in a CDN. The obtained results are presented in Fig.5.20. Fig.5.20i shows the link usage for the protocol management, both ESLR advertisement messages, and server-router communication messages. As the figure displays the protocol management consumed about 0.003% of the links and that is about 3000Kbps in a 1Gbps link. Compared to the simulation done in the Planet Lab network (see Fig.5.14), this is a clear increase in link usage. One reason is that, in this experiment, the link usage denotes both route advertisements messages and server-router communication messages. The other reason is, as pointed out in Chapter 4 Section 6, the ESLR is still a chatty protocol. Therefore, the link usage is high compared to the planet lab experiment. Again, due to the current network links are in Gigabit scale, and the link bandwidth is growing rapidly, the link usage is a tradeoff to which the benefits the users and the servers are getting.

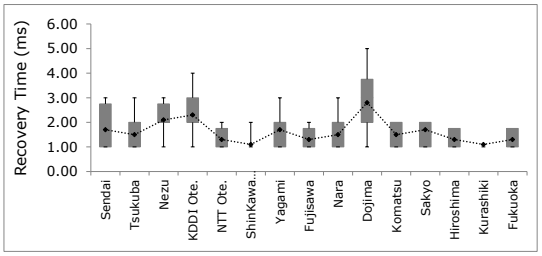
Resource usage for control plane messages The next experiment was conducted to measure the protocol overhead for the network routers. The overhead is estimated by calculating the number of packet process for defined time interval and calculating the service rate of the router accordingly. The obtained results are plotted in Fig.5.20ii. As displayed in the figure, a node averagely processes about 170B data per second. It is observed that the protocol places an above-average burden on some routers, e.g., Nezu, Dojima, and NTT Otemachi. The first reason is that the coordination activities used for route discovery and route recovery are also consumed some amount of router. The next reason for this behavior is, as explained in [96], route summarization has not yet been implemented on ESLR. Therefore, the sizes of the route tables and the sizes of the protocol messages increase. Using route summarization techniques such as variable length subnet masking can effectively solve this problem.



(I) Link usage for protocol management



(II) Protocol overhead for nodes



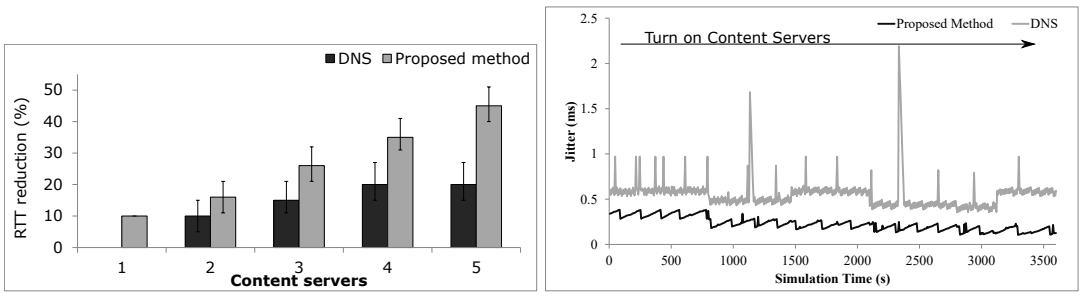
(III) Route recovery time

FIGURE 5.20: Resource usage for control plane messages

The route recovery time was measured as the third experiment of finding the effectiveness of ESLR for the proposed network. Some links were randomly disconnected during the simulation and measured the route recovery time in each router. The obtained results are plotted in Fig.5.20iii. According to the figure, the average router recovery time of ESLR is about 2.5 ms. The main reason is that ESLR maintains backup routers and ESLR users fast triggered update messages to notify the link breakdown situations. Moreover, as the ESLR uses coordinated update messages such as route pull and push techniques, the route recovery time is fast in ESLR. The main reason for this evaluation is that the high availability is one primary motivation of CDN. However, even though the CDNs maintain high availability using their servers if the underneath packet network does not maintain the network recoveries accordingly, CDNs then can't guarantee the high availability. Thus, in the proposed architecture, ESLR maintains a high availability by collaboratively using both ISP and CPs conditions.

User experience The user experience was quantified by measuring the percentage of download latency reduction and the jitter reduction similar to the planet lab experiments. Obtained results are printed in Fig.5.21. Similar to the experiment conducted in Planet Lab network, the RTT reduction was measured by increasing number of servers in the topology. Each experiment performed five times, and average results were used for the analysis. According to the Fig.5.21i, as the number of content server increases, in the collaborative architecture, the RTT decreased by up to 45%. The reason for this behavior is that, when new content servers were introduced in the network, the SoRs were able rapidly to adapt the topology changes with the aid of the server-router communication protocol. Thus, the SoRs were able to recommend new content servers to the end-users without depending on the hierarchy of DNS servers. Moreover, since the content based packet redirection is also enabled for the experiment, the SoRs were able to redirect users to the minimal busy servers even the packets are not destined to the appropriate server.

When compared to the results obtained using the PlanetLab network, which is presented in Fig.5.14, the RTT reduction and the rate of RTT reduction is high in the ns-3 simulation, 10% more reduction when using five content servers. The main reason is the ESLR. The SoRs selected the minimal busy



(I) RTT reduction (II) Jitter reduction

FIGURE 5.21: User experience evaluation

servers, and the SoRs navigate the content through the minimal delay paths, which consists of minimally congested routers and less busy links, using the ESLR. Moreover, gateway SoRs used the route metric to find the best server for the user requests. Consequently, the collaborative approach was able to reduce the effective download time in a CDN.

The next experiment was to measure the jitter. Unless in the Planet lab experiment, one client was selected in this experiment and the jitter was measured at that client by turning on servers at random time intervals. The obtained results are presented in Fig.5.21ii. According to the figure, the proposed method displays averagely 40% of jitter reduction when compared with DNS-based redirection for the entire simulation period. The main reason for this behavior is that the edge SoRs were able to redirect the users to the least-busy servers by using the minimal delay paths. Therefore, SoRs successfully navigated the content by avoiding the congested links and routers. In our experiments, we simulated a streaming service, and hence, we can conclude that the proposed method can be used effectively for streaming applications, which is a primary application of network data sharing.

Compared to the experiment done in the Planet lab simulator (see Fig.5.14), the jitter reduction is about 40% low when all five servers are turned on. The reason is, as pointed out previously, the harmony between the server selection and the network path selection. In the proposed method, working as an intermediary. SoRs collected network state information using ESLR and used those data to find minimal delay service point and navigate the content to and from that service point using the minimal delay network paths.

Behavior of the DNS Since one of the motivations of the proposed method is to leverage the DNS-based redirection by reducing the time spent on DNS resolution, the SoRs intercept the DNS request messages and recommend the DNS to the clients. The DNS recommendation process can be interpreting as resolving the DNS within the local network because gateway routers can be considered as a part of the local network. Before analyzing the DNS behavior of the simulation environment let's consider an analysis of the DNS behavior in Hiroaki Nishi laboratory. The experiment was conducted between 12:30 to 15:15 on 25th April 2016. Author measured the number of DNS requests received by the laboratory DNS server and number of DNS requests forwarded out of the laboratory gateway router. The obtained results showed that about 14% of the DNS traffic are passed to the ISP network within 45 minutes of the time period.

TABLE 5.9: Behavior of the DNS messages

	DNS-based method	Collaborative method
Resolved with in Local network	60 % (Approx.)	85% (Approx.)
Passes to the ISP	38% (Approx.)	15% (Approx.)
TTL	20s - 200s (pre-defined As per Table 5.4)	10s - 40s
Resolution time	Beyond 200ms	Less than 20ms

Now, let's analyses the DNS behavior of the simulation environment using the results presented in Table 5.9. According to the table, in the proposed network, 85% of the DNS server queries are resolved within the local network, either from the local DNS or the SoR. The reason is that, when Local DNS misses the DNS query, the SoR between the end-user and the provider network was able to recommend the IP addresses to the local DNS without forwarding the DNS queries to the public network. Consequently, the DNS server resolution time was reduced to the order of milliseconds. SoRs missed 15% of the DNS requests due to protocol management, i.e., delete server-state information records upon expiration. However, those missed DNS queries were able to resolve the IP address by contacting the Auth. DNS, which is similar to the regular DNS resolution process. This suggests that the collaborative method successfully maintains consistent data delivery by solving the hierarchical DNS resolving problem.

Response the changes occur in network and servers As stated previously, in the CDNs, the content servers are frequently changing according to the connectivity and the server loads. However, in the DNS-based redirection method, users are unable to adjust according to the server changes due to the inherited limitation of the DNS protocol. To analyses how the collaborative method reacts when there are server changes, the thesis work continued the experiments to demonstrate the effectiveness of ISP-CDN collaboration for server change situations. The analysis assumed the extreme conditions by turning ON and OFF the content servers. In each experiment, the redirection time and RTT were measured for a randomly selected user. Simulation scenarios and the obtained results are given in Table 5.10.

According to the table, in the beginning, the proposed method waited about 200 ms to identify the main server. The reason is that the DNS cache was miss at both local DNS and user edge SoR, and thus, end-user had to use hierarchical DNS resolution process to find the appropriate content server. However, results given in Table 5.10 suggest that the proposed method required on average 64 ms to recognize the changes and redirect the end-users to content servers. The reason is that the SoRs were able to collect up-to-date server-state information and use this information to suggest IP addresses for the end-users without being strongly dependent on the hierarchical DNS server resolution process. Consequently, as presented in Table 5.10 shows, 85% of the DNS server queries were resolved within

TABLE 5.10: Adapt to the changes occur in CDN

Execute at	Event	Time taken for redirection	RTT
0th Second	Start the simulation and the Main server	0.2s	0.07s
100th Second	Start Content Server 1	0.05s	0.012s
200th Second	Start Content Server 2	0.04s	0.006s
300th Second	Stop Content Server 1	0.02s	0.024s
400th Second	Start Content Server 1	0.01s	0.006s

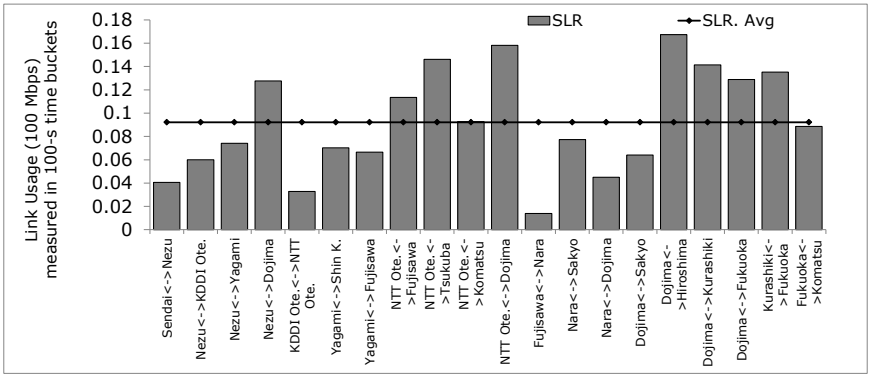


FIGURE 5.22: Network resource utilization

the local network, and thus, the DNS server resolution time reduced to the order of milliseconds. Implicitly, SoRs are successfully working as an intermediary to make a collaboration between ISP and CDNs, and hence, SoRs were able to use the ISP-CDN collaboration to suggest server changes to the end-users within small time scales. This simulation scenario explains that, if the CDNs can maintain an infrastructure that their reverse proxy servers to advertise the server state information to the gateway router, the CDNs can improve the content navigation by reducing the dependency of hierarchical DNS server resolution.

Network resource utilization The Final experiment was conducted to evaluate the network resource utilization of the collaborative approach. As explained in Chapters 1 and 2, the network resource utilization is contingent of network path selection, and network path selection is a contingent of the traffic matrix of the CP and the network devices states. Consequently, as the final experiment, the thesis study measured the network links usage for both data and protocol message navigation. Obtained results are plotted in Fig.5.22.

According to the figure, the average utilization of the links for data transfer is approximately, 10 Mbps in the proposed network. Thus, the proposed approach was able to exploit almost every link in the topology. Unless the ordinary IGPs and the ordinary TE approaches, i.e., label switching, the proposed method calculated the route matrix by considering both network device states and CPs traffic matrix. Therefore, the network path selection does not depend on the capacity of the links for route metric calculation. Nonetheless network path selection is also not optimized to one parameter such as link matrix. Subsequently, if a router identifies that a network path (i.e., a route) is busy and the propagation time is high, the router quickly switches to the backup path, which is the next best route available to reach the destination.

5.7 Discussion

The content navigation problem is the combination of service point selection and network path selection. If we elaborate it further, there are four main steps involved in the content navigation process.

1. Find the service location for a user
2. Route user's data request packets to the service location using the best path

TABLE 5.11: Summery of proposed method and the regular method

Criteria	Proposed collaborative method	Regular method
Find service location	Gateway SoR by leveraging DNS-based RR and content-aware packet redirection	Resolving iterative hierarchical DNS servers
Distance to the place which determines the service location	1 hop from the user	Averagely 10 – 18 hops from users
Time to determine the service location	Below 20 ms	Beyond 200 ms
Find the best server	Using the same step of finding the service location	Reverse proxy server or Auth. DNS server
Distance to the place which determine the Server	1 hop from the user	Almost at the data center
Time to determine the server	Below 20ms	Hierarchical DNS resolution time + time to reach the datacenter + server selection time
Criteria that determine the best server	Servers with low service time and minimal delay network path	Mysterious, [14] stays based on connection monitoring data
resource usage to collect the parameters for server selection	Medium	Low
Time used to update users about the infrastructure changes	Fast	Slow (depend on the TTL value of the DNS)
Network path selection	Consider both CPs traffic matrix and ISPs network states to determine the minimal delay path	Do not consider
Network resource usage for effective path selection	High	Low (all ISPs uses OSPF and ISIS and those protocols are low in resource usage)
Scalability	Moderate (because use of SoR are still under research and development)	High (because no protocol dependency and support almost every application)
Network resource utilization	High	Do not consider

3. Select the best server

4. Route the reply packets back to users

As explained in Chapter1, when CDNs are deployed within ISP networks, above steps above are influencing each other, and it is necessary to consider all steps as a single step. In essence, CP's traffic matrix influence for the network path selection and the traffic matrix should be a parameter for network path selection. On the other hand, network path selection should be a parameter of service point selection because even the servers are not busy but, if the network is congested, the content navigation will not display optimal results. The main motivation of the thesis study is to use gateway routers to maintain a communication platform to build a collaboration between ISP and the CPs. Thus, use the collaboration to strengthen the content navigation by selecting the best server to the subscribers and route them using the minimal delay path.

For the convenience of understanding the benefits and the limitations of the proposed method, the study is summarized in Table 5.11. The table presents a comparison related to the above mentioned four criteria when the CDN uses the proposed collaborative approach and when CDNs uses typical content navigation approach, i.e., consider DNS-based service location selection and ISP's network path selection as separate tasks.

The proposed collaborative approach uses basics of CDN concept to implement the collaboration platform, and hence, enhance the content navigation.

1. Use gateway routers of the CPs to collect server state information is similar to the reverse proxy concept, which is the basic building block of the CDN. Instead of caching the content, the proposed method used to store the server state information.

2. Intercept DNS messages and recommending the best server using gateway routers is literally similar to the forward proxy concept. Instead of fetching and storing the content, the proposed method leverages the DNS resolution by intercepting the hierarchical DNS resolution.

3. Intercept the data requests and forward the data packets to the best location of service is similar to the interception proxy service. This method is similar to the application layer redirection.

Subsequently, the dissertation proposes to use gateway routers to perform one or more methods mentioned above and enhance the content navigation by proposing a collaboration between ISP and CDN.

The collaborative approach is proposed to enhance the content navigation process by leveraging the DNS-based redirection and selecting network paths using CP's traffic matrix as a parameter. As a matter of fact, the proposed method can be work with any CDN infrastructure. However, it is indisputable that packets, e.g., end-user data requests and DNS queries, should be processed at edge routers in the proposed network. Therefore, routers should have sufficient processing power and memory to ensure

good performance of the proposed network. According to [32, 60, 61], routers with high-performance processors and in-network storage are already available in the market. Specifically, as an ongoing area of research, SoRs are improving and will be available in commerce shortly. Also, edge computing [153] and network virtualization [154, 155] are also gaining momentum among researchers and vendors. Consequently, in the short term, ISPs can lease network slices to CDNs and CDNs can use SoRs as an intermediary to strengthen the ISP-CDN collaboration.

As presented in Table 5.11, in the DNS-based service point selection approach, first users required to select the reverse proxy server using the DNS resolution. Then send the request to the reverse proxy servers. Finally, the reverse proxy server selects the best server based on some hidden criteria. However, in the proposed collaborative approach, both "service place determination" and "best server determination" are combined as a single step. The reason is that the SoRs maintain server-state information database and the table records have the less busy servers. Therefore, SoRs can send the users to the best service location irrespective of the place of the reverse proxy. In essence, the proposed collaborative architecture brings the service point selection and server selection location to the subscriber's neighborhood, i.e. gateway router. This approach significantly reduces the limitations of the DNS-based server selection and service point selection method.

Apart of the SoR, many routers in the market have the capability of analyzing packets up to L7 information and possess in network storage modules [32, 60, 61]. Such routers are also capable of providing the services similar to the SoR. Therefore, the proposed system can be implemented into the CDNs, specially Telcos, using above routers as well. Kamiyama et al. stated that ISP-CDN cooperation is the key to providing faster data delivery [152]. Furthermore, [15, 27] show that the traffic matrix should be considered as a parameter for the network path selection [82]. Subsequently, the results of this study show that, by placing SoRs at the edges of ISP networks, the SoRs can be used to leverage DNS-based CDN redirection to achieve end-user redirection in small time scales. Also, ESLR displayed effective network resource utilization by contemplating ISP and CDN state information as the parameters for network path selection. Therefore, the development of SoR [111, 156, 157] ensures that the proposed architecture will be a successful business model, which can be used by ISPs and CDNs to guarantee their subscribers are downloading data from the best server via the minimal delay path.

In the first prototype implementation, SoRs do not intercept TCP connections to provide on the fly TCP redirection because TCP is a stateful protocol, and the cost of protocol management is high. Research groups, including commercial vendors and the research community, are investigating possible solutions to intercept and redirect a TCP connection from the routers. For example, Hitachi presented a "WAN accelerator" in [33], and Cisco provided "multipath TCP" in [158]. Further, Surton et al. proposed "Man-in-the-Middle TCP recovery" in [147], and the authors suggested TCP redirection from the routers. Assuming that future CDNs and ISPs use such technologies, SoRs can be used to successfully enhance the on the fly TCP redirection.

Chapter 6

Conclusion and future vision

6.1 Conclusions

This work proposed the use of edge routers of CDNs to enhance content navigation. Content navigation is defined by two methods: 1) find the service points for users and 2) find network paths to route users to the selected service point. Edge routers work as intermediaries to collect network state information of ISPs and content server state information of content providers. The collected information is then used by SoRs to select less busy servers for the users and route users using minimal delay network paths. SoRs are placed at the edges of the CDNs as gateway routers of both end-users and content providers. The main role of the content providers' edge SoRs is to collect the server state information and advertise that information among the other SoRs in CDNs using the SLRouting protocol. In addition, all of these SoRs gather the network state information by accessing the routing tables of the SLRouting protocol. In the meantime, the main role of the user edges' SoRs is to select the best server based on the server state information and the minimal delay network path to reach the selected server. SoRs achieve those two targets by 1) leveraging DNS-based user redirection and 2) performing on-the-fly dynamic packet redirection according to the application data of the packets. They leverage DNS-based user redirection by intercepting DNS request messages and selecting the minimal delay server that has the requested URI. Instead of approximating the best service point as in DNS-based user redirection, SoRs use both network and content server state information to determine the best server that matches the user's requirement. Moreover, dissertation proposed a method to calculate the TTL values of the DNS messages using the server's utilization and the packet waiting time instead of guessing the TTL values according to the connection monitoring details. In essence, SoRs reduce the overhead of DNS packets through a hierarchy of name servers and provide rapid connection setup and rapid adaptations to the changes of the server's status.

One of the main motivations of SoRs is to analyze the packet streams for application layer information and to store that information in databases to provide user services. Consequently, dissertation used

that feature to reach the second achievement, on-the-fly packet redirection. In the proposed method, SoRs analyze the data request packets for content ID and refer to server state information databases to find the best server that matches the request. SoRs then appropriately change the destination of the data packet if the packet is not addressed to the best available servers. The proposed method used piggybacking technology to reduce the overhead of analyzing both reply packets and packets that are already analyzed and redirected. The ToS field of the IPv4 header is used to indicate the piggybacking packets.

However, the SoR introduced in [34, 35] continues to be a topic of research and development; its software and hardware components are currently being developed to create a complete router [157, 157]. Consequently, dissertation implemented a software SoR using both ns-2 and ns-3. The implemented software SoR was tested to to determine its features. The tests included performing DPI to analyze data packets and determining five tuples, analysis of the application layer data, and the redirecting of the packets according to both five tuples and application layer data. The SoRs were evaluated to examine its operation on the ns-3 simulator using two ISP topologies, ASN1221, and ASN3967. The results showed that the SoRs perform DPI with an average of 12% memory usage and 12% processor usage for a Linux machine with an Intel i7 processor and 8 GB of memory. The implemented SoR module is publicly available in [38].

Furthermore, dissertation designed and implemented the SLRouting protocol using ns-3. The implemented protocol was tested to check its functionality and the resource usage for its protocol messages. The SLRouting protocol was tested using two ISP topologies, ASN1221, and ASN3967. The protocol showed that it selects accurate network paths compared with the shortest path provided by both OSPF and RocketFuel link metrics (calculated using Dijkstra's algorithm). With the possibility of 80%, the minimal delay paths provided by the SLRouting protocol are shorter than the shortest path calculated by using the delay information of rocket fuel networks. In addition, the SLRouting protocol reduced the end-to-end propagation time by approximately 30% compared to the OSPF routing protocol. Moreover, the SLRouting protocol was able to utilize 90% of the network resources to navigate the content through the ISP networks. However, the above advantages came at the cost of using 30 Kbps more link bandwidth compared to OSPF. The latest version of the SLRouting protocol is available in [141] for those who are interested in its study, upgrade, and further development.

Finally, the collaborative approach was tested for content navigation in terms of user experience, server utilization, resource usage for control messages, network resource utilization, and fast adaptation to the changes of the servers' status using two prototype implementations. The prototypes implemented on the ns-3 simulator using WIDE network topology information showed that, compared to DNS-based end user redirection, the collaborative approach was able to reduce the RTT by approximately 20% percent. Moreover, the jitter was reduced by approximately 3040%. The server usage of the collaborative approach is approximately 0.6 times that of the DNS-based user redirection. The SLRouting protocol displayed that the link recovery time is approximately 2.5 ms in the simulated experiment. When the

system was analyzed for the behavior of the DNS messages, the collaborative approach was able to resolve more than 85% of the DNS traffic within the local network. Moreover, the DNS resolution time was reduced to less than 20 ms, and this behavior supported fast user redirections and fast connection setups. It is a fact that the DNS protocol does not have a method to notify the server states to the users unless the users contact the DNS servers when the TTL expires. However, the collaborative approach was able to overcome that limitation and redirect the users according to the changes of the servers' status within small time scales such as less than 20 ms. Nonetheless, the above advantages came at a cost of 3000 Kbps in a 1 Gbps link for control messages. Thus, the link usage can be reduced by using route compression technologies such as VLSM and table compression technologies such as Patricia tree algorithms. In conclusion, this proposed method shows the possibility of a new business model that strengthens the ISP–CDN collaboration to enhance content navigation by using edge routers of the CDNs. Consequently, the proposed method enables both ISPs and CDNs to select the least busy servers for their customers and route them using the minimal delay network paths.

6.2 Future vision

CDNs continue to evolve while the SoR is also evolving as a complete hardware appliance capable of providing user services effectively. Open Pluggable Edge Service (OPES) [159] will be the cutting-edge technology of next-generation CDNs, Dynamic Delivery Networks (DDN) [47, 160]. OPES is provided to plug the content services into the edges of the networks and enhance the user services from the edges of the network. Nonetheless, Akamai claims that it has the technology to place its service boxes at the edge of the Internet and deliver services to its subscribers [161]. However, the technology behind Akamai is a secret, and the company is not revealing proprietary design. However, IETF is particularly discussing this topic, and standards are currently being drafted to implement the OPES [162, 163]. The dissertation proposes using SoR as the edge router to enhance content navigation, which is similar to the topic covered by the OPES. The OPES consists of OPES processors, which should be capable of using both ISP and CDN properties to provide user services. This means that the SoR will be a possible candidate for the OPES processor because it has already been implemented and tested to provide similar services. Therefore, one of the foremost future visions of the dissertation is to propose the SoR as the OPES processor and implement the OPES standards using the OPES Internet Draft (ID) [159, 163].

The current prototype implemented in the dissertation is proposed to select service points based on the processing delay of the servers. However, the CDN networks select servers according to the redirection policies such as best server and nearest server. Although CDN providers do not release their redirection policies for proprietary reasons, policy-based redirection provides the flexibility of using several parameters for service point selection. Thus, the proposed method would be updated to select service points based on the redirection policies. Subsequently, as the future vision of the policy-based service

point selection, the dissertation proposes to calculate the redirection policies using the following as the parameters:

1. Processing power of the servers
2. Memory of the servers
3. Storage size of the servers
4. Available network bandwidth of the Internet link
5. Utilization of the servers

Moreover, the parameters mentioned above will be used in the OPES framework to select the service points and redirect the users' requests to the service points based on the redirection policies.

CDNs and ISPs are functional networks that are already working at a production scale. Thus, the dissertation proposed a collaborative framework using simulation implementations and merely conducted quantifiable evaluations in terms of user performance, service point selection performance, and network resource utilization. However, qualitative evaluation is required to standardize the proposed ISP–CDN collaboration framework. This requires deploying the system in a real ISP network and testing it using actual data. One of the main requirements of deploying the system in a real ISP is a mutual agreement with an service provider. That is possible by signing a non-discloser agreement (NDA) with ISPs and CDNs. Therefore, as one of the main foreseeable future vision of this dissertation, it is impeccable to sign a NDA with ISPs and CDNs. For an example, both WIDE [148] and JGN-x [149] networks can be used to initiate the dissertation work in real network. Consequently, with a trustworthy NDA, the proposed architecture will be implemented as the future OPES processor, which will be implemented using the SoR. Thus, ISPs and CDNs can stress to their subscribers that the subscribers are redirected to the servers via the minimal delay paths so that the subscribers can fetch their required content quickly, reliably, and efficiently.

Appendix A

ESLR and SRC header descriptions

Note that: all below header structures were designed and implemented by assuming IPv4 addressing scheme.

A.1 ESLR advertisement header

```
COMMAND:  Command specifies the action of ESLR message carries out
          |0|0|0|0|0|SRC|KAM|RU|
     SRC : Use for the Server-Router Communication messages
     KAM : Use for both Hello and Keep Alive messages
     RU  : Use for both periodic and triggered Route Update messages

RU_COMMAND: The type of the route update messages
          |0|0|0|0|0|0|RQ|RE|
     RQ  : Route Request message
     RS  : Route Response message
   * RU_COMMAND has to be set as "0" for both SRC and KAM messages

REQ_TYPE: For request messages, i.e., RQ, following bit combination can be
     configured to request route entries from the neighbors
          |0|0|0|0|0|ET|NE|OE|

     ET  : ENTIRE TABLE
     NE  : NUMBER of ENTRIES.
          Requesting number of entries indicate in the NoE field
     OE  : ONE ENTRY

   * REQ_TYPE bits are set to 0xff of the response, i.e., RS, messages. The
     assumption made was the MTU is 1500Bytes.

NoE:  Number of Entries that the advertisement message carries.
```

```
    Number of messages that the advertisement message is carrying is calculated
    using
    following formula.
#RUMs = |(MTU - sizeof(IPv4) - sizeof(UDP) - sizeof(ESLR))/sizeof(RUM)|

ADV_OPTIONS: Advertisement option type
        |FT|P|T|C|D|0|0|0|
    FT  : Fast Triggered Update Messages
    P : Periodic Update Messages
    T : Triggered Update Messages
    C : Connected Routes
    D : Disconnected Routes

AUTH_TYPE: Authentication type

AUTH_DATA: Authentication data

RUM: Route Update Message entries
```

A.2 ESLR RUM header

```
SEQ#: Sequence number of the route update message

METRIC: The delay to reach the destination

ROUTE_TAG:  Route tag is implemented for the future versions
      of the ESLR route updates. However, the current version
      uses the ROUTE_TAG filed to indicate poisoned routes
        |0|0|0|0|0|0|C|D|
    C : Connected Routes
    D : Poisoned Routes

NETWORK ADDRESS: The destination network address

NETMASK: Netmask of the destination network address
```

A.3 ESLR KAM/Hello header

```
COMMAND:  Command of the message
    Hello : Hello messages. This message type is used to initiate
          the neighbor discovery process.
    KAM   : Keep Alive Messages. The message type is used to
          periodically send the status checking messages among the neighbors.
```

```
IDENTIFIER: A randomly generated initialization ID. The initialization ID is used
      between tow neighbors to initiate the neighbor discovery process.

AUTH_TYPE:  Type of the Authentication used
    0x01: Plain_Text
    0x02: MD5
    0x03: SHA
  Note: This implementation used Plain_Text authentication only.

AUTH_DATA:  Data or a phrase used for the authentication
  Note: This implementation used only Plain_Text passwords.

NEIGHBOR_ID:  Unique ID that considered as the ID of an router
  Note: Neighbor ID calculation:  32bit hash of (AS# + if0 IP + mask)

GATEWAY:  IP address of the sending interface

NETMASK:  Network mask of the sending interface
```

A.4 SRC header

```
Seq#: Sequence number of the route update

Mue:  Average incoming packet rate

Lambda: Average service rate

N-bit:  Average measurements of a network network, i.e, data center

S-bit:  Average measurements of a Server

SERVER_ADDRESS: IP address of the server or the network address

NETMASK: Network prefix
```

Bibliography

[1] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, pp. 230–265, 1937.

[2] J. C. R. Licklider and W. E. Clark, "On-line man-computer communication," in *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, ser. AIEE-IRE '62 (Spring), 1962, pp. 113–128.

[3] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. (2016, July) Brief history of the internet. [Online]. Available: http://goo.gl/D2gLfI

[4] M. S. Pierre, J. Fullton, K. Gamiel, J. G. B. Kahle, J. Kunze, H. Morris, and F. Schiettecatte. (2016, July) Wais over z39.50-1988. [Online]. Available: https://goo.gl/iVd5UV

[5] S. J. Lukasik, "Will we consider ourselves better off?" *IEEE Internet Computing*, vol. 4, no. 1, pp. 47–49, 2000.

[6] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Albert, "The internet gopher protocol (a distributed document search and retrieval protocol)," Internet Requests for Comments, RFC Editor, RFC 1436, March 1993.

[7] J. Postel and J. Reynolds. (2016, July) File transfer protocol (ftp). [Online]. Available: https://goo.gl/6Z7Ssf

[8] D. E. Comer, *Computer Networks and Internets*, 6th ed. Pearson, 2014.

[9] D. Connolly and L. Masinter, "The 'text/html' media type," Internet Requests for Comments, RFC Editor, RFC 2854, June 2000. [Online]. Available: https://goo.gl/7jKQJi

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1," Internet Requests for Comments, RFC Editor, RFC 2068, January 1997. [Online]. Available: https://goo.gl/hQJpeC

[11] M. Yevstifeyev, "The 'tn3270' uri scheme," Internet Requests for Comments, RFC Editor, RFC 6270, June 2011. [Online]. Available: https://goo.gl/xFt6Ix

[12] I. L. Stats. (2016, July) Google search statistics. [Online]. Available: http://goo.gl/nz44XD

[13] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A model for content internetworking (cdi)," Internet Requests for Comments, RFC Editor, RFC 3466, February 2003. [Online]. Available: https://goo.gl/uaAOuL

[14] M. Hofmann and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice (The Morgan Kaufmann Series in Networking)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[15] M. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," in *University of Melbourne, Technical Report GRIDS-TR-2007-4*, 2007.

[16] (2016, July) Content delivery summit. [Online]. Available: http://goo.gl/33JHZ7

[17] Akamai. (2016, July) Akamai. [Online]. Available: https://goo.gl/LVDkXz

[18] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[19] Facebook. (2016, July) Facebook, inc. [Online]. Available: https://goo.gl/TmfVEZ

[20] B. Niven-Jenkins and R. van Brandenburg, "Request routing redirection interface for cdn interconnection," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-cdni-redirection-19, July 2016. [Online]. Available: https://goo.gl/Qvse6P

[21] P. Mockapetris, "Domain names - implementation and specification," Internet Requests for Comments, RFC Editor, STD 13, November 1987. [Online]. Available: https://goo.gl/33SoUx

[22] J. Moy, "Ospf version 2," Internet Requests for Comments, RFC Editor, STD 54, April 1998. [Online]. Available: https://goo.gl/0OFNVV

[23] D. Oran, "Osi is-is intra-domain routing protocol," Internet Requests for Comments, RFC Editor, RFC 1142, February 1990. [Online]. Available: https://goo.gl/QPac2k

[24] Cisco. (2016, July) Enhanced interior gateway routing protocol. [Online]. Available: http://goo.gl/2Hxhr7

[25] A. Bruno and J. Kim, *CCDA Exam Certification Guide (CCDA Self-Study, 640-861), Second Edition*. Cisco Press, 2003.

[26] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren, "Quantifying the benefits of joint content and network routing," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 243–254, Jun. 2013.

[27] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs, "Enabling content-aware traffic engineering," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 21–28, 2012.

[28] R. Aber. (2016, July) Network security basics. [Online]. Available: http://goo.gl/3st1Jy

[29] S. Higginbotham. (2016, July) In a distributed world cache is king. why routers are becoming the new server. [Online]. Available: https://goo.gl/bVDvOv

[30] Cisco. (2016, July) Cisco ace 4700 series application control engine appliance. [Online]. Available: http://goo.gl/qCBrhe

[31] ——. (2016, July) Cisco ace application control engine module. [Online]. Available: http://goo.gl/n6PZO0

[32] ——. (2016, July) Session border controller - cisco 7600 series sbc. [Online]. Available: http://goo.gl/bST5b3

[33] Hitachi. (2016, July) Hitachi to launch wan accelerator family for dramatically faster data transfer among global offices. [Online]. Available: http://goo.gl/8M1WlY

[34] K. Inoue, D. Akashi, M. Koibuchi, H. Kawashima, and H. Nish, "Semantic router using data stream to enrich services," in *International Conf. on Future Internet Technologies (CFI08)*, June 2008.

[35] K. Inoue and H. Nishi, "Semantic router using data stream to enrich services," Presentation Material, Keio Univ., Presentation, 2009. [Online]. Available: http://goo.gl/GmHnnV

[36] J. Wijekoon, E. Harahap, and H. Nishi, "Service-oriented router simulation module implementation in ns2 simulator," *Procedia Computer Science*, vol. 19, pp. 478 – 485, 2013.

[37] J. Wijekoon, R. Tennekoon, E. Harahap, and H. Nishi, "Service-oriented router module implementation on ns-3," in *Proceedings of the 7th International Conference on Simulation Tools and Techniques*, ser. SIMUTools '14, 2014.

[38] J. Wijekoon. (2016, July) The sor module implementatin for ns-3 simulator. [Online]. Available: http://goo.gl/1qPE4V

[39] Cisco. (2016, July) Software defined networking (sdn). [Online]. Available: http://goo.gl/Kv0SPD

[40] M. Turnbull. (2016, July) Why do i need two load balancers? [Online]. Available: http://goo.gl/khwJfB

[41] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[42] Sandvine. (2016, July) Global internet phenomena report. [Online]. Available: https://goo.gl/tpw2Cn

[43] Y. Abraham. (2016, July) What happens in an internet minute? how to capitalize on the big data explosion. [Online]. Available: http://goo.gl/YQXN6j

[44] F. OBI. (2016, July) Broadband performance. [Online]. Available: https://goo.gl/pLF6pI

[45] M. Rabinovich and O. Spatschek, *Web Caching and Replication*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[46] W. Jeong, G. Kim, and C. Kim, *An Efficient Backup Path Selection Algorithm in MPLS Networks*. Springer Berlin Heidelberg, 2005, pp. 164–175.

[47] Mirro-Image. (2016, July) Introducing the next generation of content delivery network dynamic delivery network. [Online]. Available: http://goo.gl/MXWCBF

[48] Incapsula. (2016, July) The essential cdn guide. [Online]. Available: https://goo.gl/2MiTIL

[49] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann, "Improving content delivery using provider-aided distance information," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10, 2010, pp. 22–34.

[50] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. M. Maggs, S. Uhlig, V. Aggarwal, and F. Schneider, "Collaboration Opportunities for Content Delivery and Network Infrastructures," *ACM SIGCOMM ebook on Recent Advances in Networking*, vol. 1, August 2013.

[51] J. van de Erve. (2016, July) Creating a forward proxy using application request routing. [Online]. Available: http://goo.gl/H1aqxD

[52] R. Droms. (2016, July) Dynamic host configuration protocol. [Online]. Available: https://goo.gl/6Cy1h6

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

[54] A. Farrel, "Chapter 2 - the internet protocol," in *The Internet and Its Protocols*, ser. The Morgan Kaufmann Series in Networking, A. Farrel, Ed. Morgan Kaufmann, 2004, pp. 23 – 77.

[55] T. Callahan, M. Allman, and M. Rabinovich, "On modern dns behavior and properties," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 7–15, Jul. 2013.

[56] J. Wijekoon. (2016, July) Implementing a dns module to ns-3 simulator based on rfc 1035. [Online]. Available: https://goo.gl/pZU6W5

[57] B. Niven-Jenkins, F. L. Faucheur, and N. Bitar, "Content distribution network interconnection (cdni) problem statement," Internet Requests for Comments, RFC Editor, RFC 6707, September 2012. [Online]. Available: https://goo.gl/3D4i40

[58] L. R. Beaumont, "Meeting worldwide demand for your content - evolving to a content delivery network," White Paper, White Paper, April 2001. [Online]. Available: http://goo.gl/RNBV5e

[59] Akamai, "Fast internet content delivery with freeflow," White Paper, White Paper, November 1999. [Online]. Available: http://goo.gl/A12sdq

[60] Cisco. (2016, July) Application extension platform (axp). [Online]. Available: https://goo.gl/Ul3afV

[61] J. Kelly, W. Araujo, and K. Banerjee, "Rapid service creation using the junos sdk," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow*, ser. PRESTO '09, pp. 7–12.

[62] J. Kim, K. Jang, K. Lee, S. Ma, J. Shim, and S. Moon, "Nba (network balancing act): A high-performance packet processing framework for heterogeneous processors," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15, 2015, pp. 22:1–22:14.

[63] Intel, "Impact of the intel data plane development kit (intel dpdk) on packet throughput in virtualized network elements," White Paper, White Paper, 2015. [Online]. Available: https://goo.gl/bDKQco

[64] ——, "Delivering 160gbps dpi performance on the intel xeon processor e5-2600 series using hyperscan," White Paper, White Paper, 2015. [Online]. Available: http://goo.gl/RhLeay

[65] K. Cho, H. Jung, M. Lee, D. Ko, T. Kwon, and Y. Choi, "How can an isp merge with a cdn?" *IEEE Communications Magazine*, vol. 49, no. 10, pp. 156–162, Oct 2011.

[66] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 181–192, 2007.

[67] Cisco. (2016, July) Cisco ace application control engine module. [Online]. Available: http://goo.gl/0gGCRN

[68] ——. (2016, July) Iox and fog applications. [Online]. Available: http://goo.gl/bmnUwi

[69] ——. (2016, July) Internetofeverything. [Online]. Available: http://goo.gl/o6cYbK

[70] Beet.TV. (2016, July) Akamai and qualcomm in pact for ultrahd streaming. [Online]. Available: http://goo.gl/K6raLy

[71] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[72] Cisco. (2016, July) Introduction to routing. [Online]. Available: http://goo.gl/E5uddX

[73] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an isp network," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 239–250, 2009.

[74] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize cdn performance," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09, 2009, pp. 190–201.

[75] E. S. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for qos-based routing in the internet," Internet Requests for Comments, RFC Editor, RFC 2386, August 1998. [Online]. Available: https://goo.gl/U4e2Ev

[76] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (te) extensions to ospf version 2," Internet Requests for Comments, RFC Editor, RFC 3630, September 2003. [Online]. Available: https://goo.gl/qYxxoC

[77] T. Li and H. Smit, "Is-is extensions for traffic engineering," Internet Requests for Comments, RFC Editor, RFC 5305, October 2008. [Online]. Available: https://goo.gl/TEv3dn

[78] K. Kompella and G. Swallow, "Detecting multi-protocol label switched (mpls) data plane failures," Internet Requests for Comments, RFC Editor, RFC 4379, February 2006. [Online]. Available: https://goo.gl/H8hi1i

[79] P. Brittain and A. Farrel. (2016, July) Mpls traffic engineering - a choice of signaling protocols. [Online]. Available: http://goo.gl/oUeqWc

[80] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. (2016, July) Requirements for traffic engineering over mpls. [Online]. Available: https://goo.gl/x5Nb5a

[81] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[82] M. Antic, N. Maksic, P. Knezevic, and A. Smiljanic, "Two phase load balanced routing using ospf," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 1, pp. 51–59, 2010.

[83] Z. Aziz, J. Liu, A. Martey, and F. Shamim, *Troubleshooting IP Routing Protocols*, 1st ed. WebEx Communications, 2012.

[84] A. Rodriguez, E. Mazurek, R. Peschke, and R. Williams. (2016, July) Networking with z/os and cisco routers: An interoperability guide. IBM. [Online]. Available: http://goo.gl/Zkoumh

[85] O. Younis and S. Fahmy, "Constraint-based routing in the internet: Basic principles and recent research," *Communications Surveys Tutorials, IEEE*, vol. 5, no. 1, pp. 2–13, 2003.

[86] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (rsvp) – version 1 functional specification," Internet Requests for Comments, RFC Editor, RFC 2205, September 1997. [Online]. Available: https://goo.gl/dPlp9e

[87] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: Provider portal for applications," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 351–362, Aug. 2008.

[88] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can isps and p2p users cooperate for improved performance?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, Jul. 2007.

[89] B. Raghavan and A. C. Snoeren, "A system for authenticated policy-compliant routing," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 167–178, Aug. 2004.

[90] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2000, pp. 519–528 vol.2.

[91] P. Brittain and A. Farrel. (Accessed:2016/Feb.) Mpls traffic engineering: A choice of signaling protocols. MetaSwitch Networks. [Online]. Available: http://goo.gl/oUeqWc

[92] A. Khanna and J. Zinky, "The revised arpanet routing metric," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 45–56, Aug. 1989.

[93] B. Jonglez, M. Boutier, and J. Chroboczek. (2016, July) A delay-based routing metric. [Online]. Available: http://goo.gl/94SrZr

[94] C. Mohit, E. Andrey, and T. Koponen. (2016, July) Data oriented network architecture (dona). [Online]. Available: https://goo.gl/I7H8Y0

[95] D. J. S. Gregg Schudel. (2016, July) Chapter 1: Internet protocol operations fundamentals. [Online]. Available: http://goo.gl/Op053G

[96] J. L. Wijekoon and H. Nishi, "Slrouting: Server link router state routing protocol design and implementation," in *Proceedings of the Asian Internet Engineering Conference*, ser. ACM-SIGCOMM AINTEC '15, 2015, pp. 1–8.

[97] J. Wijekoon, R. Tennekoon, E. Harahap, and H. Nishi, "Introducing a distance vector routing protocol for ns-3 simulator," in *Proceedings of the 8th International Conference on Simulation Tools and Techniques*, ser. SIMUTools '15, 2015, pp. 38–46.

[98] S. ISHIDA, S. HARASHIMA, H. KAWASHIMA, M. KOIBUCHI, and H. NISHI, "An efficient technique of information extraction processing in packet data management infrastructure," *IEICE technical report. Computer systems*, vol. 109, no. 474, pp. 309–314, mar 2010.

[99] Y. Nishida and H. Nishi, "Implementation of a hardware architecture to support high-speed database insertion on the internet," in *WorldComp'12*, 2012.

[100] M. Oyamada, H. Kawashima, and H. Kitagawa, "Continuous query processing with concurrency control: Reading updatable resources consistently," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 788–794.

[101] H. Yamaki and H. Nishi, "An improved cache mechanism for a cache-based network processor," in *WorldComp'12*, 2012.

[102] J. L. Wijekoon, E. H. Harahap, R. L. Tennekoon, and H. Nishi, "How can a service-oriented router merge with cdn?" *IEEJ Transactions on Electronics, Information and Systems (C)*, vol. 136, no. 8, pp. 1–8, 2016.

[103] K. Masuda, S. Ishida, and H. Nish, "Website-transverse recommendation application based on contents captured by network router," *EICE Tech. Rep*, 2012.

[104] J. Wijekoon, R. Tennekoon, E. Harahap, and H. Nishi, "Service-oriented router module implementation on ns-3," in *Proceedings of the 7th International Conference on Simulation Tools and Techniques*, ser. SIMUTools '14, 2014.

[105] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, "Seven years and one day: Sketching the evolution of internet traffic," in *INFOCOM 2009, IEEE*, 2009, pp. 711–719.

[106] H. YAMAKI, Y. NAKAMURA, K. TAKAGIWA, K. MATSUI, and H. NISHI, "High-speed decompression architecture of compressed http streams for the internet routers," *IEICE TRANSACTIONS on Communications (Japanese Edition)*, 2015.

[107] K. Ikeuchi, J. Wijekoon, S. Ishida, and H. Nishi, "Gpu-based multi-stream analyzer on application layer for service-oriented router," in *Embedded Multicore Socs (MCSoC), 2013 IEEE 7th International Symposium on*, 2013, pp. 171–176.

[108] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340.

[109] Snort. (2016, July) Snort home page. [Online]. Available: http://goo.gl/pFhxWw

[110] J. Peng, H. Chen, and S. Shi, "The gpu-based string matching system in advanced ac algorithm," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, 2010, pp. 1158–1163.

[111] K. Takagiwa and H. Nishi, "Local trend detection from network traffic. using a topic model and network router," in *WorldComp'15*, 2015.

[112] E. Harahap, J. Wijekoon, R. Tennekoon, F. Yamaguchi, S. Ishida, and H. Nishi, "Distributed algorithm for router-based management of replica server in next-cdn infrastructure," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*, 2013, pp. 266–272.

[113] J. Sawada and H. Nishi, "Hardware acceleration and data-utility improvement for low-latency privacy preserving mechanism," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 499–502.

[114] R. Tennekoon, J. Wijekoon, E. Harahap, H. Nishi, E. Saito, and S. Katsura, "Per hop data encryption protocol for transmission of motion control data over public networks," in *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*, 2014, pp. 128–133.

[115] R. Kubo, T. Shen, T. Togoshi, K. Inoue, H. Nishi, M. Tadokoro, K. I. Suzuki, and N. Yoshimoto, "Service-oriented communication platform for scalable smart community applications," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 3594–3599.

[116] DARPA. (2016, July) The network simulator - ns-2. [Online]. Available: http://goo.gl/v7f7Cx

[117] Riverbed. (2016, July) Opnet became part of riverbed. [Online]. Available: http://goo.gl/H3BxOJ

[118] Boson. (2016, July) The cisco network simulator, router simulator and switch simulator. [Online]. Available: http://goo.gl/9JMdsW

[119] J. Ahrenholz, "Comparison of core network emulation platforms," in *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, 2010, pp. 166–171.

[120] gns3. (2016, July) gns3. [Online]. Available: https://goo.gl/JX6Eyq

[121] ns 3. (2016, July) ns-3. [Online]. Available: https://goo.gl/bDgCF4

[122] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[123] ns 3. (2016, July) Routing overview. [Online]. Available: https://goo.gl/k4IWxq

[124] J. Wijekoon. (2016, July) Introducing a packet bufferto ns-3 nodes. [Online]. Available: https://goo.gl/B9pz1w

[125] D. Medhi and K. Ramasamy, "3 - routing protocols: Framework and principles," in *Network Routing*, D. Medhi and K. Ramasamy, Eds. San Francisco: Morgan Kaufmann, 2007, pp. 56 – 106.

[126] A. Farrel, "Chapter 5 - routing," in *The Internet and Its Protocols*, A. Farrel, Ed. Burlington: Morgan Kaufmann, 2004, pp. 115 – 248.

[127] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind akamai: Inferring network conditions based on cdn redirections," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1752–1765, 2009.

[128] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[129] N. Spring, R. Mahajan, and T. Anderson, "The causes of path inflation," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03, 2003, pp. 113–124.

[130] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE*, vol. 3, 1995, pp. 2129–2133.

[131] "Chapter 4 - network design problem modeling," in *Routing, Flow, and Capacity Design in Communication and Computer Networks*, M. P. Medhi, Ed. San Francisco: Morgan Kaufmann, 2004, pp. 105 – 149.

[132] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves, "A loop-free extended bellman-ford routing protocol without bouncing effect," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 224–236, Aug. 1989.

[133] U. of Washington. (2016, July) Rocketfuel: An isp topology mapping engine. University of Washington. [Online]. Available: http://goo.gl/XcYnHp

[134] A. K. Kloth, *Advanced Router Architectures*. CRC Press, Inc., 2005.

[135] J. D. C. Little, "Or forum—little's law as viewed on its 50th anniversary," *Oper. Res.*, vol. 59, no. 3, pp. 536–549, May 2011.

[136] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet interdomain traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, Aug. 2010.

[137] K. Masuda, S. Ishida, and H. Nishi, "Cross-site recommendation application based on the viewing time and contents of webpages captured by a network router," in *The 14th International Conference on Internet Computing and Big Data*, 2013.

[138] J. Garcia-Lunes-Aceves, "Loop-free routing using diffusing computations," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 1, pp. 130–141, 1993.

[139] J. Jaffe and F. Moss, "A responsive distributed routing algorithm for computer networks," *IEEE Transactions on Communications*, vol. 30, no. 7, pp. 1758–1762, July 1982.

[140] G. T. Heap and L. Maynes, *CCNA Practical Studies*. Cisco Press, 2002.

[141] J. Wijekoon. (2016, July) Server link router state routing protocol for ns3. West Lab, Keio Univ. [Online]. Available: https://goo.gl/Oi6f3g

[142] D. Savage, D. Slice, J. Ng, S. Moote, and R. White. (2016, July) Enhanced interior gateway routing protocol draft-savage-eigrp-00.txt. Cisco. [Online]. Available: https://goo.gl/ju1zcG

[143] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02, 2002, pp. 231–236.

[144] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)," Internet Requests for Comments, RFC Editor, RFC 2326, April 1998. [Online]. Available: https://goo.gl/oHXdyk

[145] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (6th Edition)*, 6th ed. Pearson, 2012.

[146] C. M. Kozierok. (2016, July) Tcp/ip routing protocols (gateway protocols). [Online]. Available: http://goo.gl/v3t1VO

[147] R. Surton, K. Birman, R. Broberg, and T. Marian. (2016, July) Man-in-the-middle tcp recovery. [Online]. Available: http://goo.gl/VN6Ph3

[148] WIDE. (2016, July) Wide internet. [Online]. Available: http://goo.gl/VAzzZC

[149] NICT. (2016, July) New genaration network testbed jgn-x. [Online]. Available: http://goo.gl/IU3dpA

[150] Princeton-Univ. (2016, July) Princeton university, planetlab. [Online]. Available: http://goo.gl/k1rgUx

[151] ——. (2016, July) Princeton university, planetlab bandwidth allocation. [Online]. Available: http://goo.gl/k1rgUx

[152] N. Kamiyama, T. Mori, R. Kawahara, and H. Hasegawa, "Optimally designing isp-operated cdn," *IEICE TRANSACTIONS on Communications*, vol. E96-B, no. 3, pp. 790–801, 2013.

[153] A. Noronha and R. M. K. O. N. Villa. (2016, July) Attaining iot value: How to move from connecting things to capturing insights gain an edge by taking analytics to the edge. [Online]. Available: http://goo.gl/fL1Nol

[154] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 181–192, Aug. 2007.

[155] ITU-T. (2015, Dec) Study group 13, future networks including cloud computing, mobile and next-generation networks. [Online]. Available: http://goo.gl/DLz6OD

[156] A. Harvath and H. Nishi, "Parallel packet processing on multi-core and many-core processors," in *The 21st International Conference on Parallel and Distributed Processing Techniques and Applications*, ser. PDPTA'15 in WORLDCOMP2015, 2015, pp. 129–134.

[157] F. Yamaguchi and H. Nishi, "High-throughput and low-cost hardware accelerator for privacy preserving publishing," in *2014 IEEE 22nd International Symposium on Field-Programmable Custom Computing Machines*, 2014.

[158] C. Pearce. (2016, July) Multipath tcp, pwning todays networks with tomorrows protocols. [Online]. Available: https://goo.gl/FkacVo

[159] A. Barbir, R. Penno, R. Chen, A. Labs, M. Hofmann, and H. Orman. (2016, July) An architecture for open pluggable edge services (opes). [Online]. Available: https://goo.gl/6q90g7

[160] Mirro-Image. (2016, July) Mirror image takes cloud services to the edge. [Online]. Available: http://goo.gl/mhJ4MU

[161] Akamai. (2016, July) Client to edge to servers to origin. [Online]. Available: https://goo.gl/yf6gOA

[162] M. Hofmann and L. R. Beaumont, "Open pluggable edge services: An architecture for networked content services," *IEEE Internet Computing*, vol. 11, no. 1, pp. 67–73, 2007.

[163] A. Rousskov. (2016, July) Open pluggable edge services (opes) callout protocol (ocp) core. [Online]. Available: https://goo.gl/Bj2Z6z

Achievements

Articles

- *Janaka L. Wijekoon*, Erwin H. Harahap, Rajitha L. Tennekoon, Hiroaki Nishi, "How can a Service-oriented Router Merge with CDN?", IEEJ Transactions on Electronics, Information and Systems (C), Vol. 136, No. 8, PP.1-8, 2016.
- *Janaka L. Wijekoon*, Erwin H. Harahap, Shinichi Ishida, Rajitha L. Tennekoon, Hiroaki Nishi, "Router-based Content-aware Data Redirection for Future CDN Systems", International Journal of Computer Network and Information Security (IJCNIS), IJCNIS Vol. 6, No. 7, PP.1-10 2014.
- *Janaka L. Wijekoon*, Erwin H. Harahap, Rajitha L. Tennekoon, Hiroaki Nishi, "Effectiveness of Service-oriented Router for ISP-CDN Collaboration", Special issue of "collaboration supports and network services for diverse and inclusive society" on Journal of Information Processing, (Conditionally accepted).
- Erwin Harahap, *Janaka Wijekoon*, Rajitha Tennekoon, Fumito Yamaguchi, Shinichi Ishida, and Hiroaki Nishi, "Modeling of Router-based Request Redirection for Content Distribution Network," International Journal of Computer Applications, Vol. 76, No. 13, pp.37-46
- R. Tennekoon, *J. Wijekoon*, E. Harahap and H. Nishi, "Prototype Implementation of Fast and Secure Traceability Service over Public Networks," The journal of Institute of Electrical Engineers of Japan, vol.11 no.S1, SUPPLEMENT 2016 issue.

Proceedings

- *Janaka L. Wijekoon*, Erwin Harahap, Hiroaki Nishi, "Service-oriented Router Simulation Module Implementation in NS2 Simulator," Procedia Computer Science, Volume 19, 2013, Pages 478-485, ISSN 1877-0509
- *Wijekoon, J.*, Harahap, E., Nishi, H., "SoR based request routing for future CDN," 2012 6th International Conference on Application of Information and Communication Technologies (AICT), 17-19 Oct. 2012
- *Wijekoon, J.*, Harahap, E., Nishi, H., "NS2 Simulation Extension for Service-oriented Router", 9th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), Santiago and Valparaiso, Republic of Chile, 12SB0094, 2012 IEICE, 1 pages.
- *Wijekoon, J.*, Harahap, E., Nishi, H., "Service-oriented Router-based CDN System, An SoR-based CDN Infrastructure Implementation on a Real Network Environment," IWFIT 2013: with congestion 39th COMPSAC, the IEEE Signature Conference on Computers, Software and Applications, Jul- 2013.
- *Wijekoon, J.*, Tennekoon, R., Harahap, E., Nishi, H., "Service-oriented Router Module Implementation on ns-3", SIMUTOOLS 2014: The 7th International ICST Conference on Simulation Tools and Techniques, Mar- 2014.
- *Wijekoon, J.*, Tennekoon, R., Harahap, E., and Nishi, H. (2015). "Introducing a distance vector routing protocol for ns-3 simulator," In SIMUTOOLS 2015, 8th EAI International Conference on Simulation Tools and Techniques.
- *J. Wijekoon*, E. Harahap, K. Takagiwa, R. Tennekoon and H. Nishi, "Effectiveness of a Service-oriented Router in Future Content Delivery Networks," 2015 Seventh International Conference on Ubiquitous and Future Networks (ICUFN), Sapporo, 2015, pp. 444-449.
- *Janaka L. Wijekoon* and Hiroaki Nishi. 2015. "SLRouting: Server Link Router state Routing Protocol Design and Implementation," In Proceedings of the Asian Internet Engineering Conference (AINTEC '15), Bangkok, Thailand.
- *Wijekoon, J.* and Nishi, H. 2016. "Implement Domain Name System (DNS) on network simulator-3 (Implement RFC 1035 on ns-3)," In SIMUTOOLS 2016, 9th EAI International Conference on Simulation Tools and Techniques.
- Erwin Harahap, *Janaka Wijekoon*, and Hiroaki Nishi, "Distributed Algorithm for Efficient Use of Replica-Servers in Content Delivery Network," 9th Asia-Pacific Symposium on Information

and Telecommunication Technologies, Santiago and Valparaiso, Republic of Chile, 12SB0094, 2012 IEICE.

- Kazumasa Ikeuchi, *Janaka Wijekoon*, Shinichi Ishida, and Hiroaki Nishi, "GPU-based Multi-stream Analyzer on Application Layer for Service-oriented Router," The 2013 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2013 in WORLDCOMP2013), pp.430-436, Las Vegas, Nevada, USA, July 22-25, (PDP3190)

- Erwin Harahap, *Janaka Wijekoon*, Rajitha Tennekoon, Fumito Yamaguchi, Shinichi Ishida, and Hiroaki Nishi, "Distributed Algorithm for Router-based Management of Replica Server in Next-CDN Infrastructure," International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC2013), pp.266-272, 10-12, Oct., 2013, Beijing, China.

- Erwin Harahap, *Janaka Wijekoon*, Rajitha Tennekoon, Fumito Yamaguchi, and Hiroaki Nishi, "Router-based Request Redirection Management for Next Generation Content Distribution Network," The 5th IEEE International Workshop on Management of Emerging Networks and Services (IEEE MENS 2013) in conjunction with IEEE Global Communications Conference, Exhibition, and Industry Forum (IEEE GLOBECOM 2013),pp.1012-1017, 9-13, Dec. 2013, Atlanta, USA.DOI: 10.1109/GLOCOMW.2013.682512

- Rajitha Tennekoon, *Janaka Wijekoon*, Erwin Harahap, Hiroaki Nishi, Eiichi Saito, and Seiichiro Katsura, "Per Hop Data Encryption Protocol for Transmission of Motion Control Data Over Public Networks," The 13th International Workshop on Advanced Motion Control, March 14-16, Yokohama, Japan, 2014, pp.133-128. DOI: 10.1109/ICTTA.2004.1307600

- Erwin Harahap, *Janaka Wijekoon*, Rajitha Tennekoon, Fumito Yamaguchi, Shinichi Ishida, and Hiroaki Nishi, "A Router-based Management System for Prediction of Network Congestion, The 13th International Workshop on Advanced Motion Control," March 14-16, Yokohama, Japan, 2014, pp.398-403. DOI: 10.1109/AMC.2014.6823315

- Rajitha Tennekoon, *Janaka Wijekoon*, Erwin Harahap and Hiroaki Nishi, "Per-hop data encryption protocol for transmitting data securely over public networks," The 4th International Symposium on Frontiers in Ambient and Mobile Systems, pp.965-972, June 2-5, 2014, Hasselt, Belgium.

- Rajitha Tennekoon, *Janaka Wijekoon*, Erwin Harahap, and Hiroaki, Nishi. "Previous Hop Data Retransmission Service for SoR-based Public Networks," The 7th International Conference on Information and Automation for Sustainability (ICIAfS2014), 978-1-4799-4598-6, Dec. 22-24, 2014, Colombo, Sri Lanka.

- Tomomichi Noguchi, *Janaka Wijekoon*, Yogendra Joshi, Minami Yoda, Hiroaki Nishi. "Shutter control for cooling air flow management in data center servers," The 42nd Annual Conference of IEEE Industrial Electronics Society, 2016, Italy.