การประเมินสมรรถนะเชิงเปรียบเทียบของนโยบายการแคชจากส่วนกลางและระดับการมองเห็นเนื้อหาในโครงข่ายแบบสารสนเทศเป็นศูนย์กลางโดยใช้ระบบทดสอบโอเพนโฟลว์

นายชันทาน เฮล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2557

COMPARATIVE PERFORMANCE EVALUATION OF CENTRALIZED IN-NETWORK CACHING POLICIES AND CONTENT VISIBILITY LEVELS IN INFORMATION CENTRIC NETWORK BY USING OPENFLOW TESTBED

Mr. Chanthan Hel

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2014

Thesis Title	COMPARATIVE PERFORMANCE EVALUATION OF CENTRALIZED IN-NETWORK CACHING POLICIES AND CONTENT VISIBILITY LEVELS IN INFORMATION CENTRIC NETWORK BY USING OPENFLOW TESTBED
By	Mr. Chanthan Hel
Field of Study	Electrical Engineering
Thesis Advisor	Assistant Professor Chaiyachet Saivichit, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Engineering

(Professor Bundhit Eua-arporn, Ph.D.)

THESIS COMMITTEE

.....Chairman

(Associate Professor Watit Benjapolakul, Ph.D.)

.....Thesis Advisor

(Assistant Professor Chaiyachet Saivichit, Ph.D.)

.....Examiner

(Assistant Professor Chaodit Aswakul, Ph.D.)

.....External Examiner

(Associate Professor Poompat Saengudomlert, Ph.D.)

ชันทาน เฮล : การประเมินสมรรถนะเชิงเปรียบเทียบของนโยบายการแคชจากส่วนกลางและระดับการมองเห็นเนื้อหาในโครงข่ายแบบสารสนเทศเป็นศูนย์กลางโดยใช้ระบบทดสอบโอเพนโฟลว์ (COMPARATIVE PERFORMANCE EVALUATION OF CENTRALIZED IN-NETWORK CACHING POLICIES AND CONTENT VISIBILITY LEVELS IN INFORMATION CENTRIC NETWORK BY USING OPENFLOW TESTBED) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.ชัยเชษฐ์ สายวิจิตร, 66 หน้า.

สมรรถนะเป็นปัจจัยหลักสำหรับการออกแบบโครงข่ายที่มีข้อมูลเป็นศูนย์กลาง(ICN) ภายใต้การสนับสนุนของโครงข่ายที่กำหนดโดยซอฟต์แวร์(SDN) นอกจากนี้การดำเนินการระหว่างนโยบายการแคชในโครงข่ายรวมศูนย์และระดับการมองเห็นเนื้อหาก็ส่งผลกระทบต่อสมรรถนะการทำงานด้วย โดยเฉพาะตัวควบคุมจะมีบทบาทสำคัญอย่างยิ่งในการประสานงานระหว่างโครงข่ายแคชและกลไกการมองเห็นเนื้อหาสำหรับโครงข่ายที่มีข้อมูลเป็นศูนย์กลางนี้ ดังนั้นวิทยานิพนธ์ฉบับนี้จึงมีแผนที่จะประเมินสมรรถนะการทำงานของโครงข่ายที่มีข้อมูลเป็นศูนย์กลางภายใต้สถาปัตยกรรมเอสดีเอ็น เมื่อพิจารณาสามกลไกที่แตกต่างกันระหว่างการแคชของโครงข่ายรวมศูนย์และการมองเห็นเนื้อหานั้นจะถูกนำมาใช้กับระบบทดสอบโอเพนโฟลว์ อีกทั้งยังมีวัตถุประสงค์เพื่อศึกษาผลกระทบของโครงสร้างโครงข่ายต่อสมรรถนะการทำงานของโครงข่าย โดยที่สามกลไกที่กล่าวถึงประกอบไปด้วยกลไกที่ไม่มีความร่วมมือ กลไกการร่วมมือระหว่างเส้นทาง และกลไกความร่วมมือรวมทั้งระบบเพื่อที่จะประเมินผลของสมรรถนะการทำงานจะมีสี่ตัวชี้วัดสมรถนะการทำงานมาใช้ สำหรับการทดสอบจะแบ่งการทดสอบเป็นสองสภาพแวดล้อมคือระบบทดสอบโอเพนโฟลว์ที่มีนโยบายการแคชและการจำลองโดยใช้โปรแกรมมินิเน็ต ในการทดสอบนั้น ผู้วิจัยแนะนำให้ใช้ มินิเน็ตในการทดสอบเพราะใช้ทรัพยากรน้อยกว่าแต่ยังคงให้ผลเช่นเดียวกับการทดสอบบนสภาพแวดล้อมโอเพนโฟลว์ในเครื่องคอมพิวเตอร์บุคคล ผลการทดสอบพบว่าในการทดสอบกับโครงข่ายขนาดเล็ก กลไกความร่วมมือทั้งระบบมีสมรรถนะสูงกว่ากลไกรูปแบบอื่นๆ แต่เมื่อโครงข่ายที่ใช้ทดสอบมีขนาดใหญ่ขึ้นกลไกความร่วมมือทั้งระบบจะแสดงข้อด้อยบางประการในด้านอัตราการส่งข้อความระหว่างชั้นควบคุมและชั้นข้อมูล และจุดคอขวดของทราฟฟิกในข่ายเชื่อมโยง นโยบายการแคชแบบไม่ประสานงานและระดับการมองเห็นของแต่ละโนดให้ผลสมรรถนะต่ำที่สุด โดยแนวทางการเลือกกลไกนั้นจะขึ้นอยู่กับการตั้งค่าจริงของผู้ดูแลระบบโครงข่ายและทรัพยากรที่มีอยู่ในระบบขณะนั้น

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2557

5570579421 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: CENTRALIZED IN-NETWORK CACHING / CONTENT VISIBILITY / OPENFLOW TESTBED / INFORMATION CENTRIC NETWORK (ICN) / SOFTWARE DEFINED NETWORKING (SDN) / PERFORMANCE EVALUATION

CHANTHAN HEL: COMPARATIVE PERFORMANCE EVALUATION OF CENTRALIZED IN-NETWORK CACHING POLICIES AND CONTENT VISIBILITY LEVELS IN INFORMATION CENTRIC NETWORK BY USING OPENFLOW TESTBED. ADVISOR: ASST. PROF. CHAIYACHET SAIVICHIT, Ph.D., 66 pp.

Performance is a key factor for designing Information Centric Network (ICN) under the support of Software Defined Networking (SDN). Moreover, the implementation of centralized in-network caching policies and content visibility levels surely affects the performance. This dissertation plans to evaluate the performance of ICN under SDN support when three different mechanisms of centralized in-network caching policy with content visibility level are applied by using OpenFlow testbed. It also aims to study the effects of network topology on the network performance. Those three mechanisms consist of non-cooperative in-network caching policy with individual content visibility mechanism, path cooperative in-network caching policy with path content visibility level mechanism and global cooperative in-network caching policy with global content visibility level mechanism. To evaluate the performance; four performance metrics including *server hit ratio*, *average hop count*, *message rate between control and data plane*, and *bottleneck link traffic* were utilized. The experiments were conducted in two experimental environments including PC-based OpenFlow testbed and emulation by Mininet. In each environment, two different network topologies were tested. The results show that each mechanism has its strengths and weaknesses. Also, the network performance corresponding to each mechanism depends on the types of network topology. While the network is simple, the global cooperative in-network caching policy with global content visibility level mechanism outperforms others. But, when the network becomes larger, this mechanism shows some disadvantages in terms of *message rate between control and data plane*, and the *bottleneck link traffic*. The non-cooperative in-network caching policy with individual content visibility mechanism performs the worst compared to others. On the other hand, emulation by Mininet is recommended to use because it utilizes less resource and gives the same results as those in PC-based OpenFlow testbed environment.

Department: Electrical Engineering Student's Signature .....
Field of Study: Electrical Engineering Advisor's Signature .....
Academic Year: 2014

ACKNOWLEDGEMENTS

First of all, I would like to show my gratitude to my advisor, Asst. Prof. Dr. Chaiyachet Saivichit. He always gives me good advices and comments for conducting this research. Without him, I cannot complete this dissertation. I would also like to thank Asst. Prof. Dr. Chaodit Aswakul for his useful comments and suggestions. He usually considers me as one of his advisees and helps me a lot. On the other hand, I also thank to Assoc. Prof. Dr. Watit Benjapolakul and Assoc. Prof. Dr. Poompat Saengudomlert for being the chairman and external committee for my thesis.

I also thank to all my friends, seniors and juniors in Network Research Group who are always helpful to me. They share me the research experiences, comments and suggestions during my study here. Especially, we are like brothers and sisters; I feel warm during my stay here and they make me feel like home here.

I, particularly, express my acknowledgement to AUN/Seed-net, JICA project for financial support during my stay here. Also, this research has been financially supported by the Special Task Force for Activating Research (STAR) Funding in Wireless Network and Future Internet Research Group, Chulalongkorn University.

Without Institute of Technology of Cambodia (ITC), I cannot be here now. I express my grateful to ITC for giving me a good opportunity to study here, Thailand.

Finally, I would like to express my profound grateful to my parents and family. They have sacrificed everything for me, without them I cannot have today.

CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Currently, the Internet usage is rapidly increasing. People need more services and applications of Internet to make their daily life easier. Hence, many companies and Internet service providers have been creating more services to maintain their market shares in this competitive world. However, the current Internet architecture based on the IP network has a lot of obstacles to deploy the new Internet services and applications; it is a host centric network. The researchers proposed another new network architecture named "Information Centric Network (ICN)". In contrast to IP network, ICN focuses on the name of content rather than the location. It is believed that the ICN might replace the current IP network architecture in the future. To make the network more transparent and easy to manage, the concept of ICN architecture under the support of Software Defined Networking (SDN) has been proposed. This network architecture could be also called ICN over SDN. This network is a centralized network. It is separated into two planes, data plane and control plane. There is one or many network controllers, situated in the control plane, control the ICN nodes that located in the data plane.

In order to design the ICN over SDN, the network performance needs to be taken into account. In-network caching is one of the main functionalities of ICN besides routing, and naming. It is the process that all the ICN nodes inside the network cache the data content in their own content storage. According to many researches, in-network caching policy has influence on the network performance. On the other hand, content visibility/searching level also affects the effectiveness of request forwarding to the destination. In ICN over SDN architecture concept, the controller is a key player to coordinate the in-network caching policy and content visibility/searching process. Nevertheless, the controller can search for the content's location and coordinate the ICN nodes to perform in-network caching with many

different manners. We believe that different levels of content searching and caching managed by the controller will cause different network performance. It is important to be aware of the network performance corresponding to various in-network caching polices and content visibility/searching levels. It helps network administrators select the best mechanism suitable for their preference and resource availability.

1.2 Problem statement

Performance is the main issue that must be considered as a key factor to design the ICN architecture. Moreover, researches show that one of many ways to improve the performance of ICN is cooperative in-network caching. It means that the ICN nodes must collaborate with each other to cache data content inside the network. Furthermore; content visibility, the ability of ICN nodes to see contents inside the network, also affects the whole network performance. Some researchers proposed the partial cooperative in-network caching policy with partial content visibility level in the network [4]. Other researchers proposed path cooperative in-network caching policy with path content visibility level that only the ICN nodes on the path from the requester to the server can do the collaboration to cache the content and share content information with each other [5], [6]. But, the work studied the effect of global cooperative in-network caching policy with global content visibility level on the network performance is uncommon. Also, most of proposed works are based on the distributed network not the centralized network [4], [5], [6]. There is neither experiment nor simulation study to compare the performance of ICN over SDN by applying different centralized in-network caching policies with different levels of content visibility by using OpenFlow testbed. On the other hand, the global content visibility level is not paid much attention to. In ICN over SDN, the controller plays a very important role to coordinate in-network caching and content searching/visibility process. At this point, three mechanisms of centralized in-network caching policy with content visibility/searching level are considered. Namely, they are non-cooperative in-network caching policy with individual content visibility mechanism, path cooperative in-network caching policy with path content visibility level mechanism and global cooperative in-network caching policy with global

content visibility level mechanism. Each mechanism is composed of an in-network caching policy and a level of content visibility/searching. In the non-cooperative in-network caching policy with individual content visibility mechanism, the network controller just only keeps the routes to forward request/interest and data/content packets in the network. The controller does not coordinate the ICN nodes to cache or look for the data content. For the path cooperative in-network caching policy with path content visibility level mechanism, the controller chooses only the ICN nodes along the shortest path from the requester to the server to cooperate with one another to cache data contents. Also, the controller can only search for location of required content storage along the same path. Finally, in the global cooperative in-network caching policy with global content visibility level mechanism, the controller has the ability to search for locations of all contents in the network and to decide to cache data packet in any ICN nodes inside the network. Furthermore, the network topology may also be influential in the network performance. By seeing problems as aforementioned, we have questions as follow:

1. Among the three mechanisms as aforementioned, which mechanism gives better performance in ICN under the support of SDN and OpenFlow?
2. Does performance in question (1) vary depending on the network topology?

1.3 Objectives

This thesis plans to evaluate the performance of ICN under the support of SDN when different mechanisms of centralized in-network caching policy with content visibility level are applied by using OpenFlow testbed. Also, it aims to observe the effects of the network topologies on the network performance.

1.4 Scope of thesis

This research focuses on the ICN's performance evaluation. Performance metrics used in this work including *server hit ratio, average hop count, message rate between control and data plane* and *bottleneck link traffic*. The research will cover and consider on the following issues:

1. Conducting experiment only in one autonomous system of ICN over SDN based on OpenFlow testbed.
2. Two experimental environments are taken into account for this research, PC-based OpenFlow testbed and emulation by Mininet.
3. Implementing the PC-based OpenFlow testbed for performing the experiment.
4. Studying only three different centralized in-network caching policy with content visibility level mechanisms in ICN under the support of SDN.
5. The content caching process in path cooperative in-network caching policy with path content visibility level mechanism and global cooperative in-network caching policy with global content visibility level mechanism is heuristic, the most popular content is cached as near as possible to the requesters.
6. Testing in two types of network topologies, cascade and three-level binary tree network topology.
7. The message content shall be sent as one entity, not to be fragmented
8. The request pattern is assumed to be known and all requesters request content with the same request pattern.

1.5 Expected outcomes and contributions

After finishing this research, we expect to get benefits as follow:

1. Evaluating the performance of ICN over SDN while applying various centralized in-network caching policy with content visibility level mechanisms by using OpenFlow testbed. Which mechanism will give the best performance corresponds to specifically given conditions.
2. Offering knowledge to the network administrators to pick the best mechanism for implementing in ICN over SDN architecture corresponding to their available resources.
3. Understanding the effects of network topologies on the ICN-based network performance.

4. Contributing the PC-based OpenFlow testbed platform for one autonomous system of ICN under SDN support dedicated only for in-network caching and content visibility for other researchers to use as the ideas for moving ICN architecture or developing other testbed under the support of SDN/OpenFlow concept which is the main component for future networking.

1.6 Organization of dissertation

The rest of this thesis is arranged as follow. **Chapter 2** describes the background of information centric network, software defined networking and ICN over SDN concept as well as OpenFlow. The literature review is also included in that chapter. The detail of the three centralized in-network caching policy with content searching/visibility level mechanisms and how to implement the PC-based OpenFlow testbed for performing the experiment are demonstrated in **Chapter 3**. **Chapter 4** evaluates the performance results obtained from the experiment while deploying three mechanisms as aforementioned. The conclusion of the whole work and the future research direction discussion are indicated in **Chapter 5**.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Background

2.1.1 Information centric network

Information Centric Network (ICN) is a new network architecture paradigm that catches attentions from researchers around the globe. This network is hoped to substitute the current IP-based network in the future. Its concept focuses on the name of content rather than content's location. It means that requesters or users can request data content by name. Also; ICN nodes, routers or switches in the network, forward packets based on the name of content. Another special feature of ICN is in-network caching; all the in-network ICN nodes have ability to cache the content passing through them. Many ICN architectures have been proposed [7], [8]. Although those ICN-based network architectures are different, they aim to the same direction by focusing on the content-centric rather than host-centric architecture. To realize this network, three main research topics, ICN's components, are being conducted such as naming, routing by name and in-network caching process [9].

2.1.2 Software defined networking

Unlike the conventional network architecture, Software Defined Networking (SDN) is a network where the control and the data plane are separated from each other [2]. Also, the network functionality and service can be programed by using software. In this network, the controller, situated in the control plane, has the ability to configure and command network components in the data plane to forward packet or apply any actions on the packets in the network. This concept makes the network more transparent. It means that the controller has the global overview of the whole network infrastructure. Furthermore, it is easy for infrastructure or service providers to manage, reconfigure or modify their services or network functionalities by just using the software in the controller. Figure 2.1 demonstrates the logical architecture of SDN.

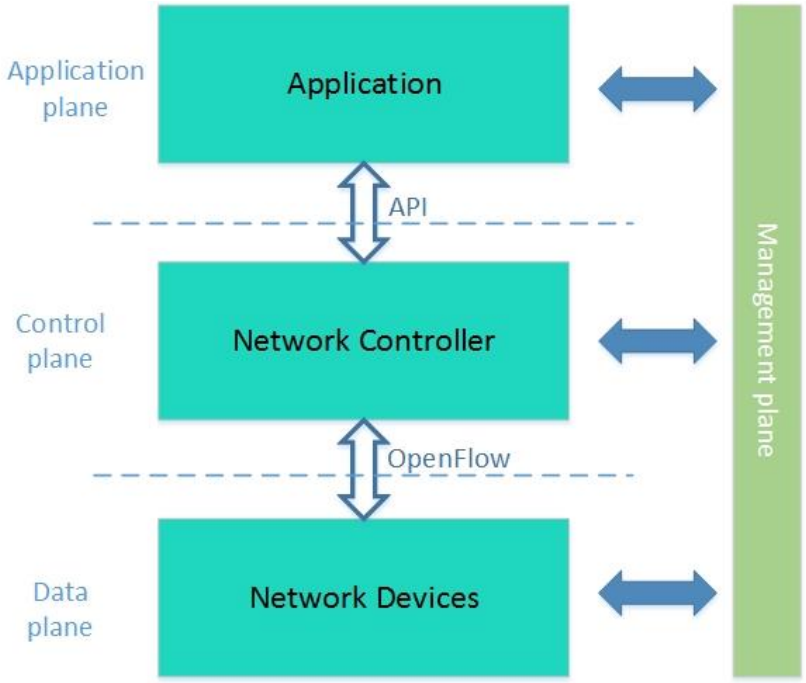

Figure 2. 1: Logical architecture of SDN [2], [3]

There are four main planes in the network: application plane, control plane, data plane, and management plane located on top of the other three planes. But, the most important thing to differentiate SDN concept from the traditional networking concept is the separation between the control and data plane.

2.1.3 OpenFlow

OpenFlow is the standard communication interface between control and data plane of OpenFlow network/SDN architecture [2]. There are three main components in the OpenFlow network; they are OpenFlow switch, OpenFlow controller and OpenFlow protocol as shown in Figure 2.2.

OpenFlow switch is a switch that supports OpenFlow. It is situated in the data plane of the network. Its role is to do packet matching and apply action on the packet based on its *flow entries* in the *flow table*. There are at least three components in the OpenFlow switch including *flow table*, *secure channel* and *OpenFlow protocol* [10]. A *flow table* contains one or several *flow entries* used for

performing packet matching. The *flow entry* consists of *rule* that is used to match against packet's header field; *action* utilized for performing the action on packet based on the rule matching; and *statistic* of the packets. Figure 2.3 demonstrates the packet's header field can be matched against flow entries in OpenFlow switch specification version 1.0.0. The *secure channel* is utilized to enable communication between the OpenFlow switch and the controller through *OpenFlow protocol*.

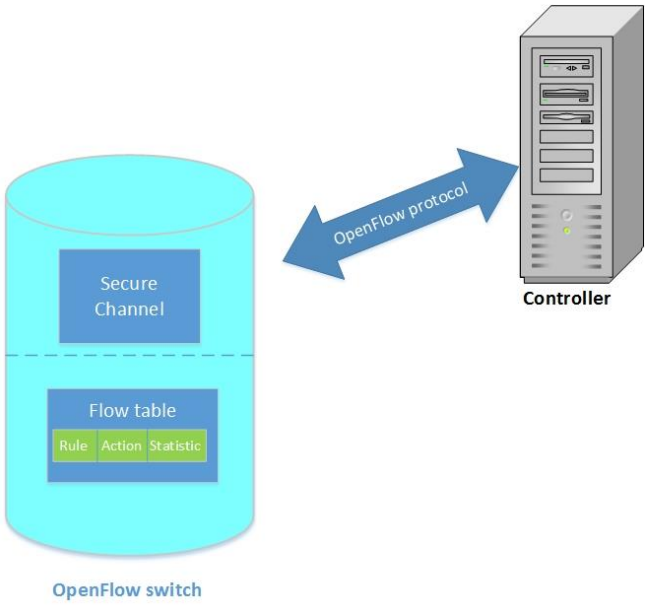

Figure 2. 2: OpenFlow network components [1]

Ingress Port	Ether src	Ether dst	Ether type	VLAN ID	VLAN priority	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP src port	TCP/UDP dst port

Figure 2. 3: Packet's header field can match against flow entries [1]

The OpenFlow controller plays an extremely important role to control the packet flow in the network. It can modify flow entries in the flow table and command the OpenFlow switch to apply action on packet.

2.1.4 Information centric network over software defined networking

The design concept of this network is to build ICN based on SDN concept and under the support of OpenFlow protocol. The ICN architecture is separated into two planes, the control and the data plane. The communication protocol between these two planes is OpenFlow protocol. The network architecture design, the operations on ICN packets and packet design etc., were previously studied and discussed in [8],[11], [9] and [12]. Figure 2.4 shows the architecture of ICN over SDN. There are at least four main hardware components comprising the controller situated in the control plane; ICN nodes, content requesters and content server, in the data plane. Again, the roles of the controller and the ICN node are similar to those of the OpenFlow controller and the OpenFlow switch respectively. The controller of this network is very crucial. It helps coordinate the packet flow in the data plane. It also keeps the whole overview of network architecture and routing information. Every mechanism, policy, e.g. caching policy, is implemented in the controller. The ICN node is responsible for performing packet matching, storing the data content and communicating with the controller. However, the requester and the server just only requests and serves the data, accordingly.

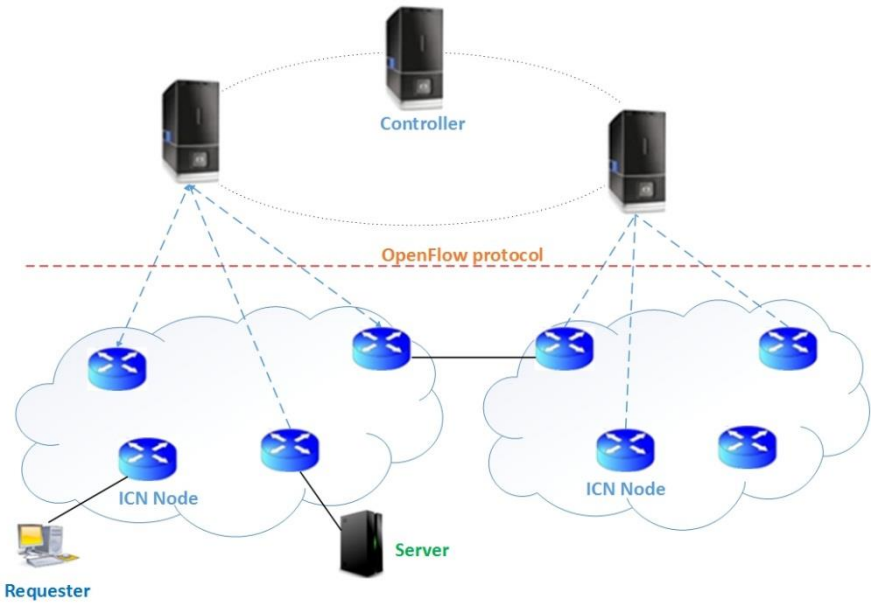

Figure 2. 4: ICN over SDN architecture

2.2 Literature review

Information Centric Network (ICN) is a new network architecture for the Internet and is currently under studying. Many researchers and research organizations have proposed several ICN architectures and concepts [7]. Performance is the key metric needed to be considered in order to design ICN. In-network caching is a hot topic for ICN's design. Lots of researchers and research organizations have been conducting many researches to improve the performance of ICN by taking the in-network caching into account. There is a consensus that the in-network caching affects the performance of ICN [5], [13]. Some researchers proposed in-network caching policies by considering the collaboration of all ICN nodes for making content caching decision; and others proposed another relevant in-network caching issues. The goal of these researches is to contribute some ideas or researching findings to improve the performance of ICN that is under studying and developing. There are several kinds of cooperative in-network caching such as partial, path, neighboring and global cooperation policy [13]. Most of the works were studied in the distributed and decentralized network. Routers in the network decide by themselves whether to cache content passed through them according to information they get from other routers [5], [4], [6].

Moreover, content visibility level also affects the ICN's performance. It related closely to the cooperative in-network caching policy. The term content visibility here refers to the ability of ICN nodes for seeing the available content in the network [13]. When the ICN node can know more available contents, the performance of network can be enhanced because the ICN node can forward the request to reach the content's location more effectively. In order to improve the content visibility in the network, the content must be advertised frequently. To do so, it requires more message overheads, bandwidth and especially it is difficult to realize the global visibility level with distributed network. If the global content visibility level is considered, the best way is to implement a centralized network where all contents are registered or updated to the global routing/resolution system [13]. This point shows that the centralized network should be adopted to improve content visibility

level. Many works considered only partial and individual content visibility [5], [4], [6]. It is difficult to find the works that took global content visibility into account for their study. Also, the works that compared the performance of ICN by applying various cooperative in-network caching policies and content visibility levels are uncommon.

Reference [14] compared the performance of ICN based on three cache placement policies. Namely, they are Least recently used (LRU), single-path and network-wide caching policies. The LRU is the manner of non-cooperative in-network caching. The single-path is a caching policy that ICN nodes along the path from the requester to the server do collaboration together to cache data content. Moreover, for the network-wide, all ICN nodes inside the network cooperate with each other to perform in-network caching [14]. This literature also used Mixed Integer Programing (MIP) to determine the location where the content should be cached in order to minimize delivery cost. But, that work was studied in decentralized manner and the ICN node can only search for data contents in its own storage. There is no global content visibility. Among the proposed ICN architectures, PUISUIT [15] is an architecture that has a Rendezvous system. The content publishers must advertise their content to the Rendezvous system. All the requests are forwarded to the Rendezvous for doing content matching [7], [15]. It follows the concept of the centralized control and the global content visibility. However, researches that used that concept to evaluate the performance of ICN with cooperative in-network caching policy are rare.

Meanwhile, Software Defined Networking (SDN) is increasingly interested by researchers. OpenFlow also plays a very important role to support the SDN concept. It can provide tools to create the testbed for performing experiments in the real network architecture [10]. SDN and OpenFlow are becoming more and more important for future networking. Many networking commercials such as Cisco, NEC, Ericsson and others, start migrating their networking products to operate with OpenFlow protocol. Cisco Company now is developing Cisco Open Networking Environment (Cisco ONE) product that is based on SDN concept. Their idea is to separate the control and the data plane of the network from each other [16], [17]. According to this fact, ICN architecture should be designed under the support of SDN

and OpenFlow. At this point; [8], [11], [9] proposed the ICN architecture based on SDN concept under the support OpenFlow protocol. The detail of how the ICN over SDN is designed in real application such as operations on packets in network, naming, routing and packet forwarding process were also indicated. Moreover, they also gave details of how the ICN can be implemented in SDN/OpenFlow testbed. This network is a centralized-control network. However, those works focused on how to migrate the whole conventional ICN into SDN/OpenFlow concept, but they did not focus much on the in-network caching yet. They did not conduct any experiments to evaluate performance of ICN by using various levels of cooperative in-network caching and content visibility. Also, the researches that evaluated network performance by deploying various levels of cooperative in-network caching and content visibility in ICN over SDN architecture are uncommon. This has left the gap that we have to do more investigation about it. However, [18] applied quite similar concept that we want to do. It aimed to study advantages and disadvantages of web cache collaboration by using centralized-control network. They evaluated the performance of the network by changing level of in-network caching policies such as no caching, independent caching and cooperative caching. The result of this work shows that the cooperative caching with centralized control gives better performance compare to independent and no caching of web page in backbone network. But the network studied by [18] is IP-based network, it is not the ICN-based. The control and the data plane were not purely separated because, at that time, there was no SDN protocol to do pure plane separation.

In our work, we want to observe and compare the performance of ICN under SDN support by applying different levels of centralized in-network caching and content visibility using SDN/OpenFlow testbed. We adopt many ideas from [11], [9], [8] in order to implement OpenFlow testbed for our experiment. We follow their ICN under the support of SDN architecture design, but we take only one autonomous system into account for our study.

CHAPTER 3

CENTRALIZED IN-NETWORK CACHING POLICY WITH CONTENT VISIBILITY LEVEL MECHANISMS AND PC-BASED OPENFLOW TESTBED

This chapter will explain the operations of three centralized in-network caching policy with content visibility/searching level mechanisms in ICN over SDN which are taken into account in the performance evaluation. Each mechanism is the combination of one centralized in-network caching policy and one level of content visibility. These three mechanisms consist of non-cooperative in-network caching policy with individual content visibility, called “non-cooperation mechanism”; path cooperative in-network caching policy with path content visibility level, called “path cooperation mechanism”; and global cooperative in-network caching policy with global content visibility level, called “global cooperation mechanism”. The main difference between these mechanisms is the ability of the network controller to look for content storage in the network (content visibility) and to command ICN nodes to perform cooperation for content caching (in-network caching policy). Moreover, this part will also describe the detail of the PC-based OpenFlow testbed implementation used for performing our experiment.

3.1 Operation of various centralized in-network caching policy with content visibility level mechanisms

3.1.1 Non-cooperative in-network caching policy with individual content visibility mechanism

Non-cooperative in-network caching policy with individual content visibility mechanism is simple; it can be called “non-cooperation mechanism”. The network controller does not coordinate ICN nodes to cache the content. Also, it does not help the ICN nodes search for any data contents. Each ICN node is responsible by itself for content caching or searching. The controller just only stores the routes to

forward request/interest and data/content packets in the network, shortest path for this work.

When a request packet arrives at one of the ICN nodes connected to the requester, it would be called *"edge node"*, this node checks the desired data content in its own cache. If the requested content is previously stored in its cache, it will reply with the data back to the requester. If there is no desired data in the cache, it will ask the controller which port this request packet should be forwarded to get the shortest path to the server. Based on the network topology, the controller can determine the shortest path from the requester to the server. The controller commands the edge node to forward the request to the next ICN node along the determined path and to store this set of action in the edge node's flow table. While the request packet reaches the ICN node 2 as depicted in Figure 3.1, the ICN node looks for the desired data packet in its content storage. In case that the desired data is found, that data packet will be forwarded back to the edge node, the ICN node that just sent the request packet to the ICN node 2. Rather, the ICN node 2 cannot find the desired data packet; it inquires the controller where to transmit that request packet to. The controller does the same action as it did with the edge node. The process is repeated for every ICN node receiving the request packet along the path from the requester to the server. The outcomes received from the controller are stored in ICN nodes. ICN nodes will know where to forward the request to without asking the controller again later. In another case that the requested data content is not stored in any ICN nodes' storage along the shortest path, the request packet will be served by the content server.

On the other hand, every ICN node along the path from the ICN node that replies the requested data to the content requester will copy every data packet. If the cache of ICN node is full, one content will be removed randomly from the cache and replaced by the new content. It is the manner of the "always" caching policy as described in [19] and the random replacement policy.

Figure 3.1 shows the sequences of operation on packets in non-cooperative in-network caching policy with individual content visibility mechanism. It is a simple example of one path that is selected by the controller composed of two ICN nodes,

one requester and one server. Let us assume that the requester requests the data content name *"D1"*. The request packet is sent to the edge node. The edge node replies with the data back if the data is stored in that ICN node (step 1 of Figure 3.1). The operation will continue to step 2 if the requested data is not kept in the edge node. The edge node inquires the route from the controller, the controller replies with a set of actions on packet and routing information back to the edge node. The edge node, then, employs action received from the controller on the request packet by sending it to the ICN node 2. If the desired data is stored in ICN node 2, the node will forward the data back to the edge node. The edge node copies the data to store in its cache and sends packet to the requester. In another case that the data is stored in the server, the request is transmitted to the server as in step 3. The ICN node 2 and the edge node will cache the data *"D1"* before sending it out.

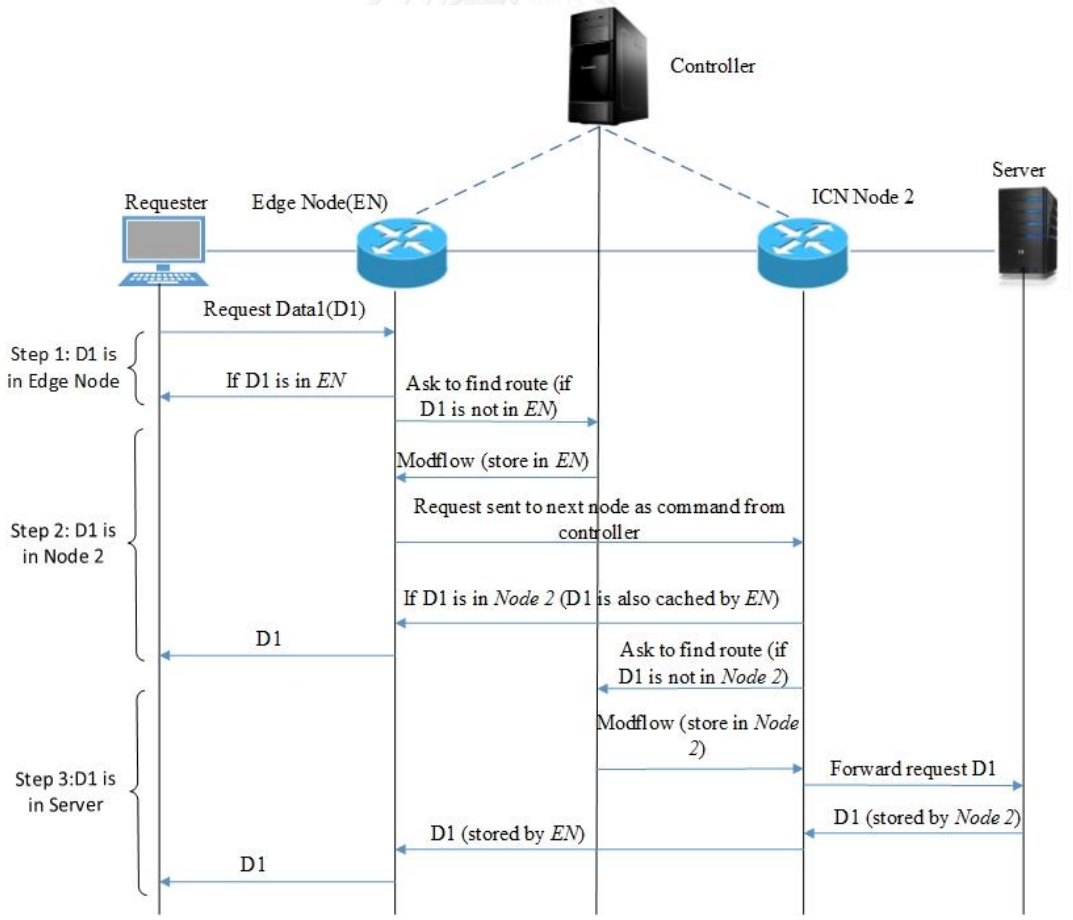

Figure 3. 1: Sequences of operation on packets in non-cooperation mechanism

3.1.2 Path cooperative in-network caching policy with path content visibility level mechanism

This mechanism is a combination of path cooperative in-network caching policy and path content visibility/searching level. The controller chooses only the ICN nodes along the shortest path from the requester to the server to cooperate with one another to cache contents/data. Also, the controller can only search or see locations of required data along the same path. This mechanism can be named "path cooperation mechanism".

3.1.2.1 Path content visibility/searching level

Contents in ICN nodes along the shortest path from the requester to the server can be seen and searched for by the controller. This is called path content visibility level. The contents' location searching process is processed by the controller when it receives the request packet's information from edge nodes.

When a request packet first reaches one of the edge nodes, that packet's information is sent to the controller. The controller, then, checks the requested data in the list obtained from the edge node. If required data packet is previously stored in the edge node's cache, the controller will instruct the edge node to reply with the data back to the requester. If the desired content is not stored in the edge node's cache, the controller will determine the shortest path from the requester to the server. Then, the controller looks for the location of the desired data content. It checks in lists obtained from other ICN nodes along the path in the direction to the server until it can find the data packet. Firstly, the controller finds out the required data in the ICN node situated one hop away from the edge node, i.e. ICN node 2 as depicted in Figure 3.2. In case that the required data can be found in the ICN node 2, the controller will instruct the edge node to modify the header field of the request packet by adding an identifier to determine that the desired data packet is stored at the ICN node 2, then, forward the request packet to the ICN node 2. In another case that the controller cannot see the data in the ICN node 2, it will continue to search for the data packet located two hops, three hops and so on, away from the edge node along the path in the direction to the server until it can find the location of

required data packet. Once the requested data is found at, for example, the ICN Node *i*, the controller will command a set of actions to the edge node to modify header's field of the interest packet by adding an identifier to define that the desired data is located in the ICN node *i* and to send the packet out. So, when ICN nodes along the path between the edge node and the ICN node *i* receive the request packet, they will check their flow tables in order to perform actions on the packet. If they cannot find any actions in the flow table, the request packet's information will be updated to the controller. The controller will then issue a command for ICN nodes to forward this packet and to store routing information in the ICN nodes' flow tables. In another case which the controller cannot find the requested data packet inside the network, it will send instead the route to forward that request packet to the server.

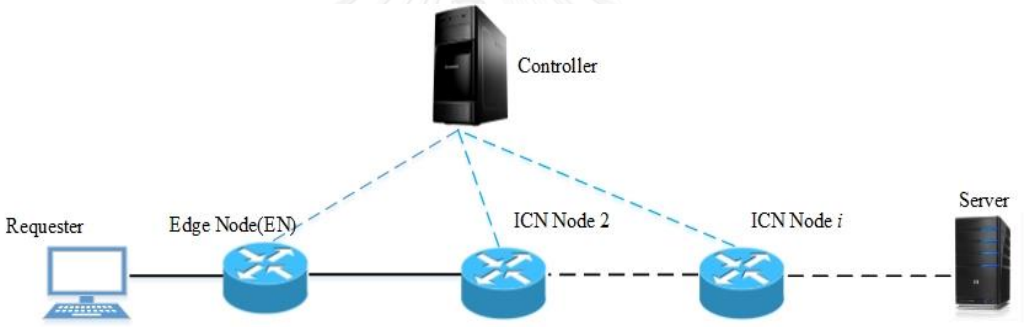

Figure 3. 2: A shortest path selected by the controller

3.1.2.2 Path content/in-network caching policy

In this mechanism, the path in-network caching policy was applied. It is the popularity-based in-network caching policy. The controller decides upon the location where the data should be cached along the same path that the request packet has passed through based on content popularity that it has collected from the edge nodes in the network. In the path in-network caching policy, the most popular contents are cached in the edge node and other ICN nodes as near as possible to the requester, along the path. The less popular contents are stored farther from requesters. This will distribute diversity of contents in the network.

The network controller classifies contents requested by the requesters in several classes: the first most popular class, the second most popular class, the third and so on. Contents of the first most popular class are cached in the edge node or the ICN node situated one hop away from edge node or two hops away and so on along the path. It depends on the popularity of that content class. The number of ICN nodes storing the first most popular class's contents is proportional to percentage of popularity of such that content class. For example, if the contents of the first most popular class are requested 50% of the total requests, so 50% percent of total ICN nodes along the path in the direction from the requester to the server can cache the contents (Figure 3.3). Note that, in this caching policy, one data content is stored in only one ICN node; there is no data duplication. The contents of the second most popular class can be stored in the last ICN node that caches the contents of the first most popular class and/or in the other upstream ICN nodes. The contents of the third most popular class are cached farther from the requester compared to the second most popular class. Content of the less popular classes are cached farther from requester. The number of ICN nodes that cache the contents of each content class is proportional to the percentage of content class popularity.

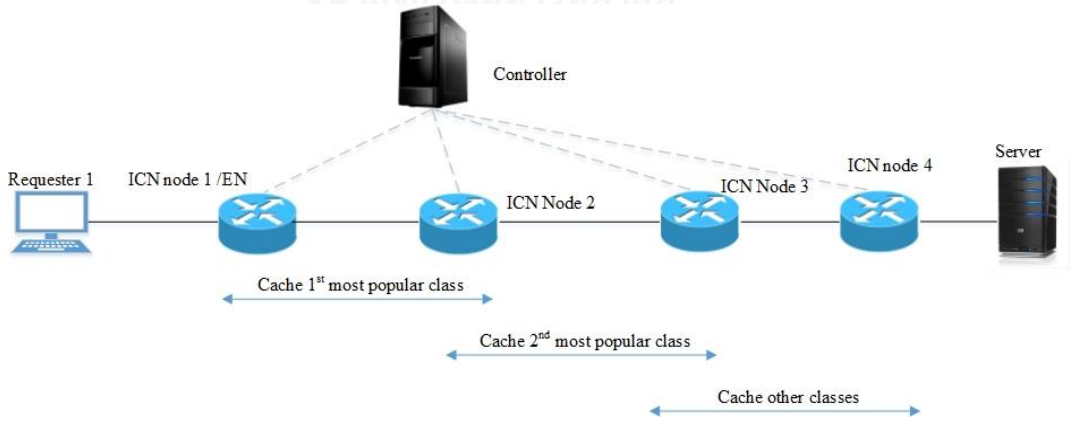

Figure 3. 3 : ICN nodes store contents in path in-network caching

When a request packet reaches the ICN node containing the requested data, its information is transmitted to the controller. The controller decides to cache the data packet in any ICN node along the same path the request packet has travelled

across depending on the content popularity. Then, the controller commands a set of actions to modify the header field of the data packet by adding an identifier to determine the location where the data content ought to be cached before sending it out. This operation is similar to the content searching process. While the data packet arrives at intermediate ICN nodes of the path, ICN nodes will forward packet according to the routes in their flow tables. In contrast, ICN nodes will transmit the data packet's information to the controller to ask for the route to forward the packet unless there is matching in their flow tables. All the outcomes from the controller are stored in ICN nodes' flow tables. On the other hand, if the data is from the server and arrives at the ICN node connected to the server, that ICN node will send packet's information to the controller. The controller will decide upon the location where that data should be cached in the same manner as aforementioned. Note that for this mechanism, the controller can only decide where to cache the data in the ICN nodes along the path from the server to the requester as predetermined by the controller during the content searching process.

In order to be easy to understand, we will demonstrate an example about this path cooperative in-network caching policy with path content visibility level mechanism. Let us assume that the shortest path chosen by the controller is shown in Figure 3.4 and the desired content C1 is stored in the ICN node 3. In this case, C1 is not popular content, so the controller decides to keep the content C1 in the same place, i.e. ICN node 3. The sequences of operation to get content C1 when there is no route to the content's location are shown in Figure 3.4.

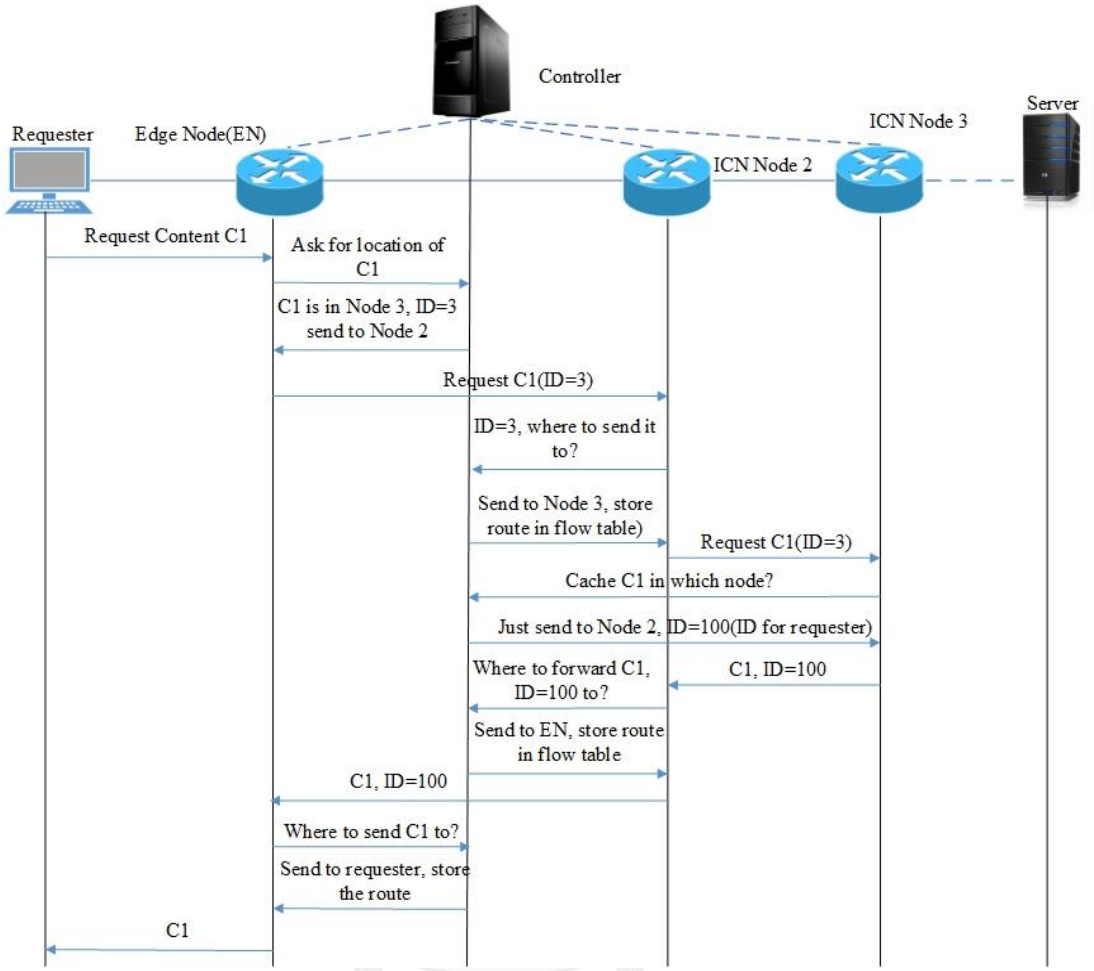

Figure 3. 4: Sequences of operation to fetch the content C1 for the first time

Once the content C1 is first requested and the request packet arrives at the edge node, the edge node sends the information to the controller for obtaining the set of action on that request. After the controller finishes the content searching process, it commands the edge node to perform actions on the request packet. The edge node modifies the header field of request packet by adding *ID = 3* to determine that the desired data content is stored in the ICN node 3. The request packet is forwarded to the ICN node 2. While the request packet arrives at the ICN node 2, the ICN node does packet matching against the flow entries of its flow table. In this case, the C1 is firstly requested, so there are no flow entries matching the request packet. The request packet information is again sent to the controller. The controller commands the ICN node 2 to forward the request packet to the ICN node 3 and to store the

routing information in the ICN node 2's flow table, i.e. *Request packet with ID = 3, Action: forward to ICN node 3.* Now, the request reaches the ICN node 3. The information is transmitted to the controller to determine the location where the data content C1 should be cached. At this time, the controller decides to keep the data content C1 in the same place, i.e. ICN node 3. So, the controller issues the command for the ICN node 3 to add *ID = 100*, the ID to identify the location of the requester, in the content C1's header field. The data is sent to the ICN node 2. The ICN node 2 and the edge node inquire the action sets from the controller to act on the data packet. The routing outcomes are store in the flow tables.

When the content C1 is requested again, the sequences of operation are manifested in Figure 3.5. Some ICN nodes, i.e. the ICN node 2 and the edge node, have already stored the routing information for forwarding the request and the data C1 in their flow tables, so the packets can be matched against the flow entries without questioning the controller again.

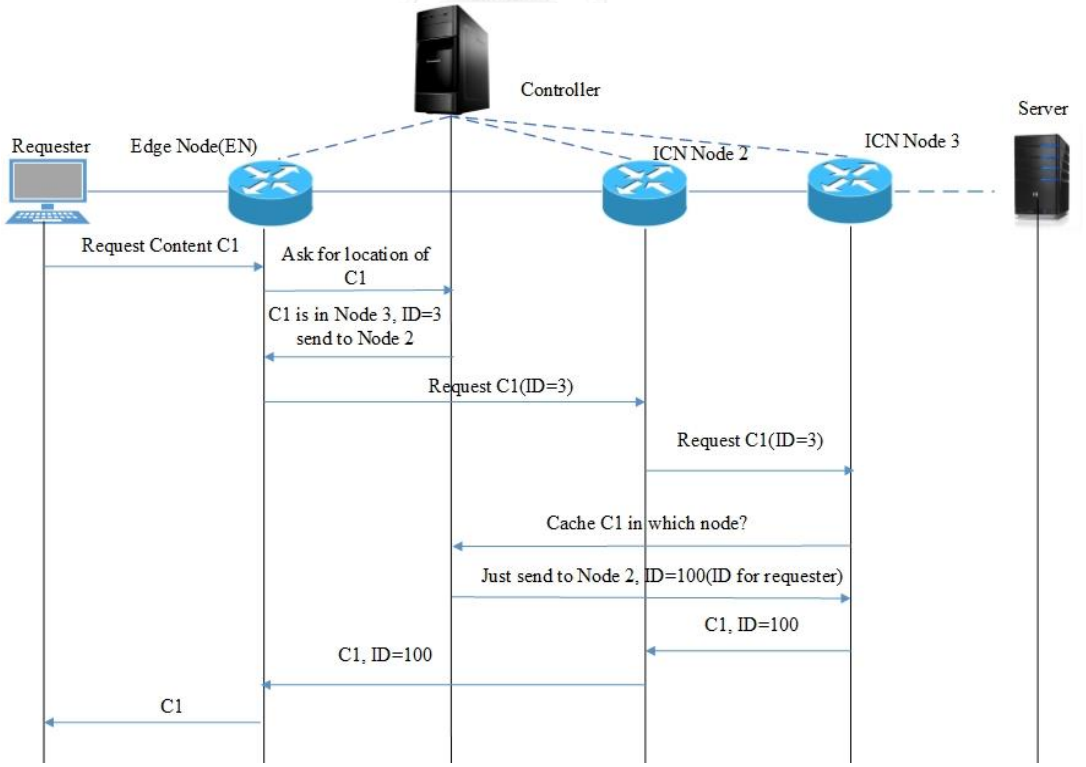

Figure 3. 5: Sequences of operation to fetch the content C1 for the later time

3.1.3 Global cooperative in-network caching policy with global content visibility level mechanism

In this mechanism, the controller has an ability to search for locations of all contents in the network and to select any ICN nodes where the data should be cached inside the network. That is why this mechanism is called global cooperative in-network caching policy with global content visibility level mechanism or in short "global cooperation mechanism". This mechanism is a combination of the global content visibility/searching level and the global content/in-network caching policy.

3.1.3.1 Global content visibility/searching level

Unlike the path cooperative in-network caching policy with path content visibility level mechanism, the controller in the global cooperative in-network caching policy with global content visibility level mechanism has the ability to search for desired data contents in every ICN node in the network (global visibility level), not just only along the shortest path.

When a request packet reaches the edge node, its information is sent to the controller. The controller then searches globally for the location of the desired data content. First of all, the controller checks for required data in the content list obtained from the edge node, and then it finds out in the lists obtained from every node situated one hop, and continues in the next two hops and so on, away from the edge node. The content searching process is finished when the controller finds the location of the data content (Figure 3.6). In case that the data is not in the network, the controller will choose the shortest path to send that request packet to the server. In another case where the data content is stored in the network but not at the edge node, the controller will calculate the shortest path from the requester to the ICN node that stores the desired data. The controller will order the edge node to put an identifier in the header field of the request packet before sending out to the next ICN node.

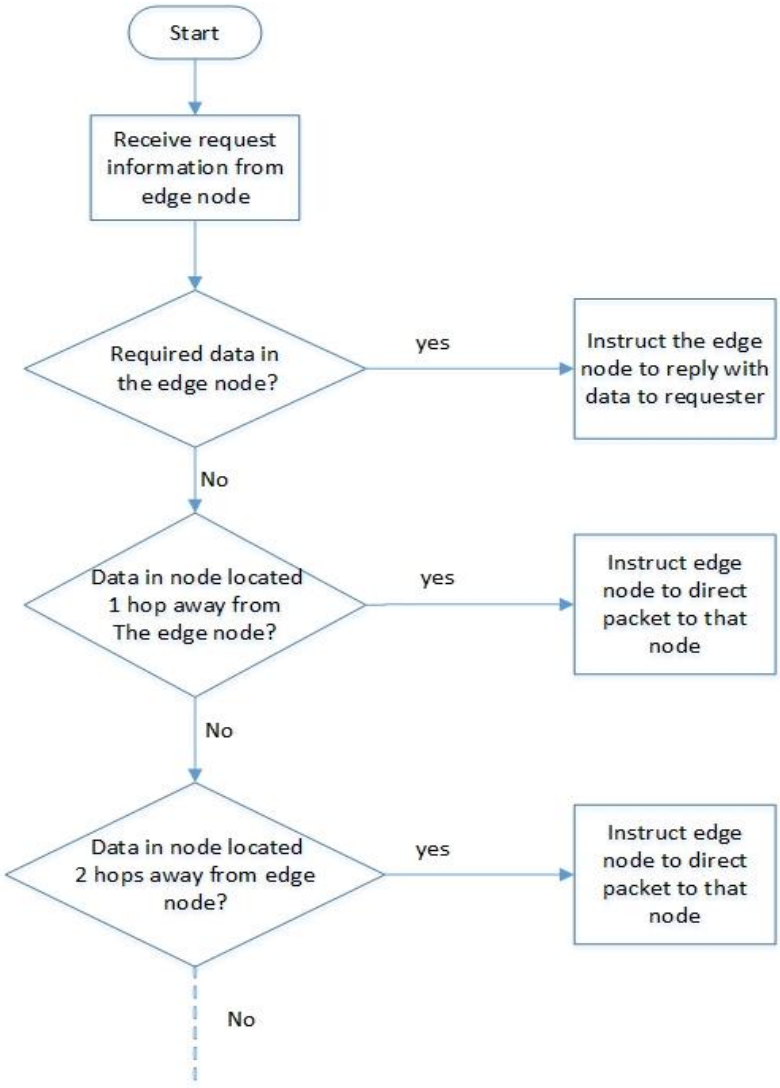

Figure 3. 6: Content searching steps in the controller

When the request packet arrives at the next node, for example ICN node 2 in Figure 3.2, the ICN node 2 then checks in its flow table whether any flow entry matches against the request packet. If it cannot see any route to forward that packet, it will send the packet's information to the controller. Then, it applies action directed from the controller on the request packet and sends the packet to next hop. The operation is repeated until the request packet reaches the ICN node that stores the required data. All the routing information from the controller will be stored in the intermediate ICN nodes between the edge node and the ICN node containing desired packet. The operations on the request packet at the intermediate ICN nodes are

similar to those of the path cooperative in-network caching policy with path content visibility level mechanism.

3.1.3.2 Global content/in-network caching policy

In this mechanism, the global in-network caching policy was utilized. It is the popularity-based in-network caching policy. The most popular contents will normally be cached near the requesters. The controller has global view of the network topology and it also has all the statistics of contents in every ICN nodes. The data content is not only cached in ICN nodes along the path from the requester to the ICN node that stores the data content, but it can also be cached in other ICN nodes in the network (global in-network caching decision). The contents are divided into several classes based on their popularity such as the first most popular class, the second most popular class, the third most popular class and so on. In this in-network caching policy, contents of the first most popular class are stored in the edge nodes and other ICN nodes nearest to most of requesters that have the same request pattern. Contents of the second, the third most popular class and so on are cached more farther from most of requesters comparing to the first, the second most popular class and so on, respectively. Numbers of ICN nodes storing contents in each class are proportional to the percentage of content class popularity. On the other hand, contents requested by a requester might be replicated to store in other ICN nodes near the other requesters also. It depends on the network topology.

When a request packet reaches the ICN node storing requested data packet, the request packet's information is transmitted to the controller. At this point, the controller selects the ICN nodes in the network to cache the data content. When the decision of content caching is done, the controller instructs the ICN node to reply with the data and forward such data content to the requester along the same path that the request packet has passed through. An identifier is attached in the data packet's header field in order to indicate which node to cache such that content. If the controller decides to cache the data packet in several ICN nodes in the network, the controller will instruct the ICN node that stores the data to copy several data packets, then send them to various directions. The controller will help ICN nodes

forward data packets to deliver to the requester and the ICN nodes that need to cache such data packet. The sequences of operation on the data packets in the intermediate ICN nodes, the ICN nodes located between the ICN node that stored the desired data and the ICN nodes that will cache the desired data, in this mechanism are same as those in path cooperative in-network caching policy with path content visibility level mechanism.

For example, if Requester 1 requests content C1 and this request arrives at the ICN node containing the desired content, i.e. ICN node 7 as depicted in Figure 3.7, the controller will decide upon the location where the content C1 will be cached. Assume that C1 is not only the most popular content for the Requester 1, but also for the Requester 4. Then, the controller decides to cache C1 in the ICN node 1 and the ICN node 4. The set of instruction is sent to the ICN node 7 to copy two packets; one is forwarded to the ICN node 1 and another to the ICN node 4 (Figure 3.8). Before sending out, the identifiers are attached to the header field of those packets to define that those packets will be cached in the ICN node 1 and 4. For example, *ID=1* and *ID=4* are attached in the data packets travelling in the direction to the ICN node 1 and 4, accordingly.

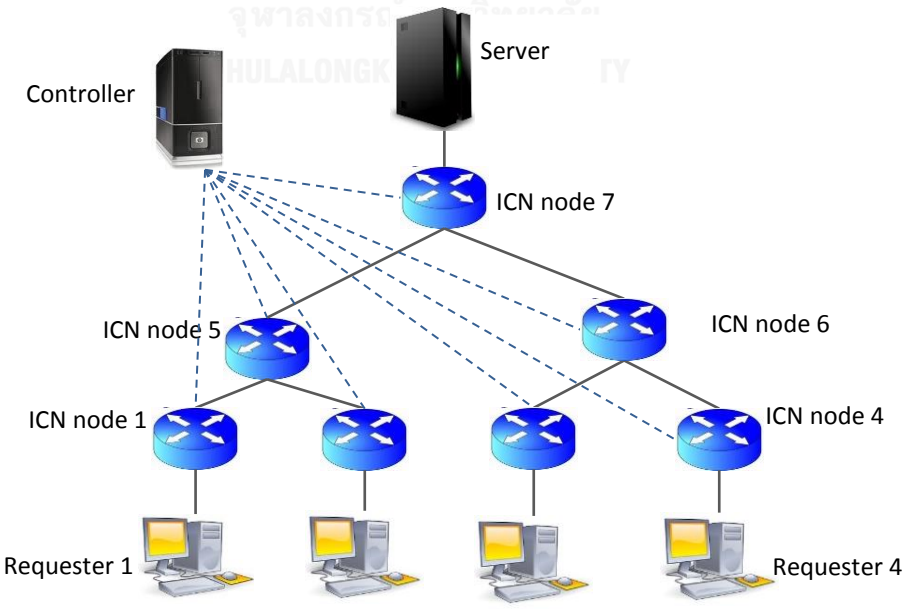

Figure 3. 7: Tree network topology

The intermediate ICN nodes, i.e. ICN node 5, will do content C1 packet matching with its flow table's flow entry. While there is no matching, the ICN node 5 will send data packet's information to the controller to ask for the route to forward this packet. The controller will give a set of route and action for performing on such that data packet C1. The routing outcome from the controller is stored in the ICN node 5's flow table. This case is for the data packet C1 that is forwarded to the ICN node 1 direction. The operation on the data C1 for the direction to the ICN node 4 is same as those to the ICN node 1 too.

The brief of operation in this mechanism is shown in Figure 3.8. Let us assume that the Requester 1 in Figure 3.7 requests content C1 and C1 is stored in ICN node 7. The controller decides to cache C1 in the ICN node 1 and 4 as being mentioned above. Also, there is no any flow entry in ICN nodes yet.

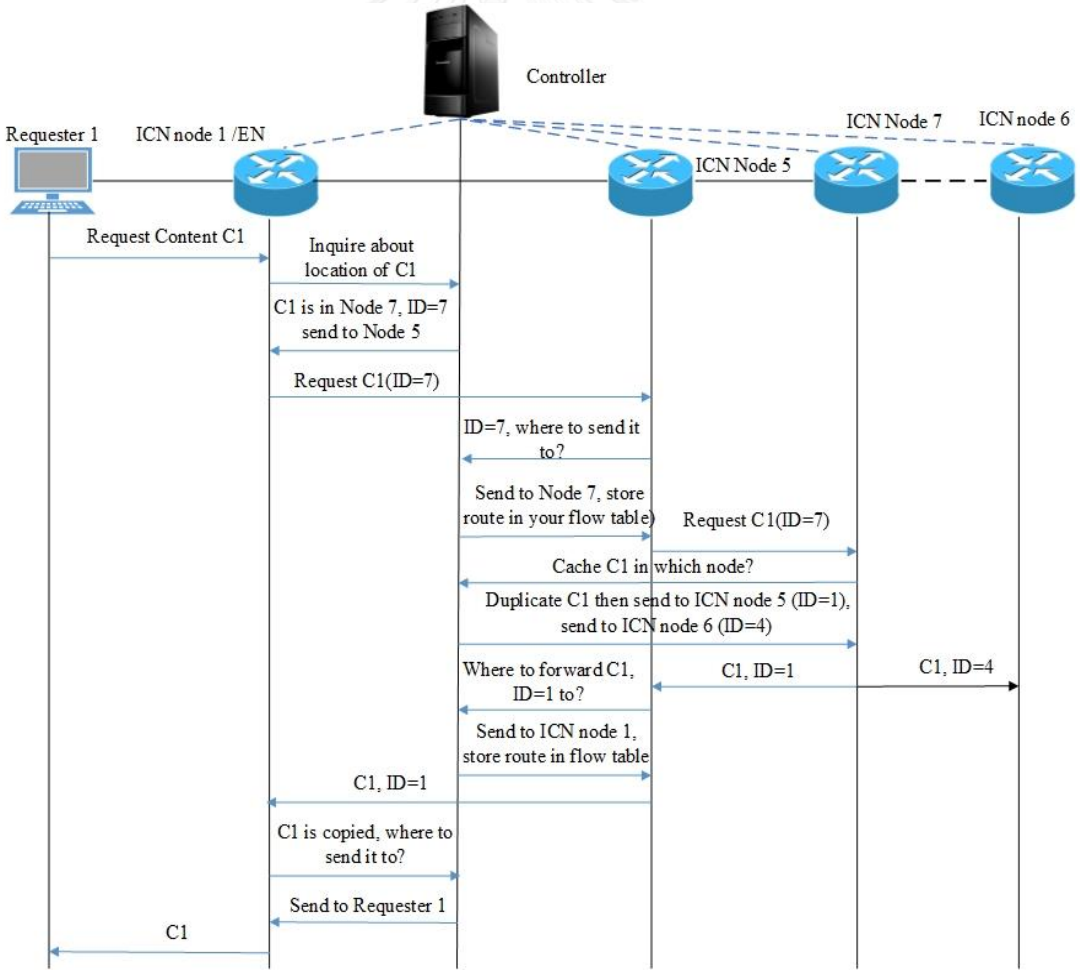

Figure 3. 8: Sequence of operation to fetch the content C1

3.2 PC-based OpenFlow testbed implementation

In the real ICN under SDN support design, OpenFlow might be one of the communication protocols between control and data plane. Every network's component must have OpenFlow interfaces. Those interfaces will be used for enabling OpenFlow protocol between ICN nodes and the controller. On the other hand, the ICN packet should be differently designed from the current IP packet. However, in this work, the network architecture and packet design are not much considered. For ICN over SDN architecture and packet design, we adopt the concept of [11], [9], [8]. In order to realize our experiment, it is very important that the OpenFlow testbed must be implemented. This part will explain in detail how the PC-based OpenFlow testbed for ICN is implemented in just only one autonomous system (AS). How the network components and the ICN packets are created, how to differentiate the name and types of ICN packets, and the operation on packets will be described in this part.

Our PC-based OpenFlow testbed is created in order to conduct experiments in small scale lab testbed to evaluate the performance of ICN as being described in the objectives of this work. Figure 3.10 is the picture of the PC-based OpenFlow testbed that is implemented in Telecommunication Laboratory, Chulalongkorn University. Figure 3.9 is one of the network topologies taken into account for this research.

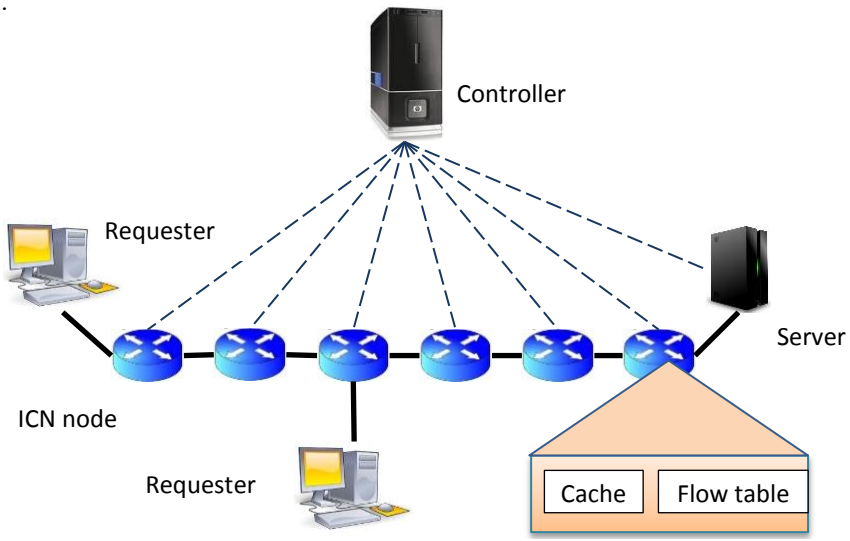

Figure 3. 9: Cascade network topology used for experiment

Figure 3. 10: PC-based OpenFlow testbed

3.2.1 Testbed component implementation

There are four main hardware components in this testbed including: ICN nodes, the network controller/controller, requesters and the content server.

3.2.1.1 ICN node

The ICN node is a network component located at the data plane in ICN under the support of SDN. Its role is to match against and forward packets based on the flow entries in its flow tables. It works as a switch or a router. Additionally, it has its own embedded cache to store the contents as shown in Figure 3.9. However, according to our implementation, the ICN node cannot store the real contents. It only updates the packet's information to the controller, and then the controller records the names of contents and stores them in its list.

To create one ICN node, a computer (PC) is utilized. That PC has 2Gbytes RAM and Celeron-2.7GHz clock speed CPU. That PC is installed with open vswitch software version 1.9 [20]. This enables the PC becomes an OpenFlow switch. USB ports are considered as the ICN node ports. In order to make the USB interfaces

become Ethernet interfaces as in the real switch, USB-to-LAN adapters are used. Some switch's ports are added to the data plane interface and another switch port is utilized as secure channel-supported interface to connect to the controller as shown in Figure 3.11.

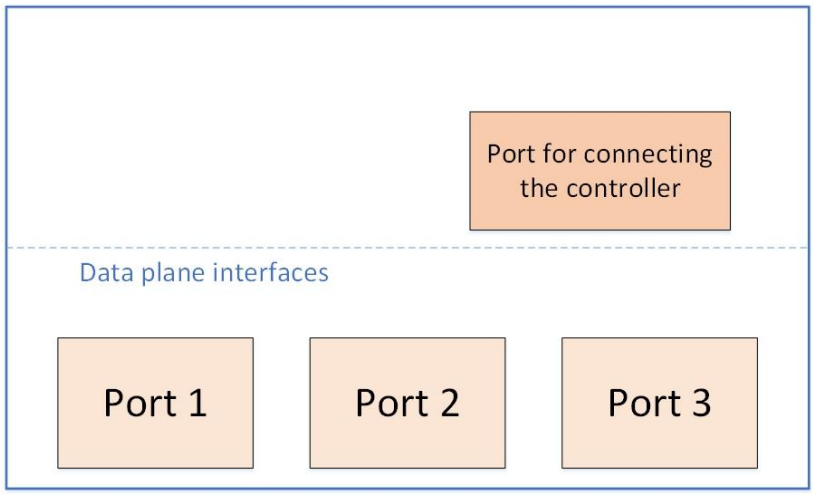

Figure 3. 11: Interfaces of the PC-based ICN node

3.2.1.2 Network controller

The network controller or the controller plays a very important role to coordinate the mechanisms mentioned in section 3.1. It keeps all routing and network topology information. Moreover, it collects and keeps all contents' information such as contents' names from every ICN nodes' caches. Based on these statistics and control coding; the controller can determine the actions on packets such as packet forwarding, packet caching or removing, etc. Also, the controller can modify the flow entries in the ICN's flow table. The code or any algorithm can be implemented in the controller. The controller's codes corresponding to each mechanism in this work are developed and run in the controller.

To build the controller; python-based OpenFlow controller software, POX [21], is installed in a core-i5-1.8 GHz clock speed computer. POX supports only OpenFlow switch specification 1.0 [1].

3.2.1.3 Content server

In the real network, the content server keeps all the contents. In case that the controller cannot find any requested contents in in-network caches, it can apply actions to direct the request packet to the content server. In this work, it is really hard to create the content server that can store real contents. We use a PC installed with open vswitch software version 1.9 as a server and it is controlled by the controller also. In this study, the payload of request and data packet is the same; so it is easy to create the server by using open vswitch, the detail of operation is demonstrated in section 3.2.4.2.

3.2.1.4 Requester

A Scapy-installed PC is used as a requester. In this work, Scapy software [22] is utilized to manipulate request packets by adding UDP source port number and source MAC address in packet's header field. The UDP source port number is used to identify the name of content and source MAC address is to recognize the request packet.

3.2.2 Types of packets

In the ICN, there are two types of packets flow in the network, the request packet and the data packet. The request packet refers to a packet that requesters generate to demand data contents in the network. The data packet is a packet which contains the data content requested by requesters. These packet types must be separated from each other by some identifiers that OpenFlow can identify. However; in our experiment, the IP packets are considered as the ICN packets. The OpenFlow protocol now can only support the IP packet type. In this testbed, the request and the data packet have the same payload. It means that one packet is one time a request packet and another time a data packet. But the difference between those two types of packet is packet's header. To identify the type of ICN packet, the tagging solution is applied in the header field of IP packet. These kinds of packet are separated by the MAC address of requesters. The request packet is a packet whose header contains the MAC address of a requester as its source MAC address no matter

where it is in the network. The data packet is a packet whose header contains a requester's MAC address as its destination MAC address. Note that, the header's field of the packet in OpenFlow network is transparent and modifiable for the controller. That why it is very easy to use any packet's header field to recognize the packet type in tagging process. On the other hand, in the testbed setup, none of the TCP/IP protocol suites such as TCP, UDP protocol is used in the data plane. The requester just only generates and injects request packets into the network; then, the packet is processed by the OpenFlow network. The IP packet is borrowed to use as the ICN packet in this case.

3.2.3 Naming process

All contents in ICN must have different names. There are two naming systems that are usually used in the ICN naming concept, flat and hierarchical naming scheme [7]. In this research, the naming scheme is not much focused. The flat naming scheme is chosen. The requester requests contents according to contents' name. In order to be easy to implement the testbed, the requesters are allowed to request contents by UDP source port number s instead of readable names as proposed in [8]. Because it can be processed in the OpenFlow network, it does not need to have hash tables/function to convert the original name field to field that can be processed by the OpenFlow network. The UDP source port number is a 16 bits header field [1], so there are $2^{16} = 65536$ different contents in the network. That number of contents will be sufficient for conducting our experiment; we need only 1000 different contents. Figure 3.12 demonstrates packet's header field to identify packet type and name of content. Let us assume that "A" is the MAC address of requester in Figure 3.9. MAC address "B" can be any address which is not conflict with MAC addresses of other requesters.

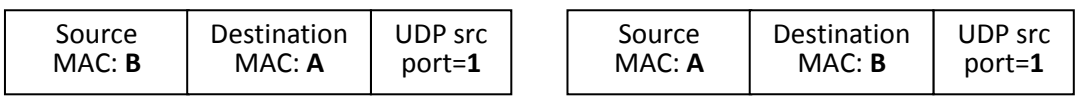

Figure 3. 12: Packet's header field to identify name and type of packet: (a) is a data packet with name 1; (b) is request packet with name 1

3.2.4 Operation on packets

This section describes the operation on ICN packets in both planes of our PC-based OpenFlow testbed. It shows how the packets flow in the testbed network.

3.2.4.1 Operation in ICN node

First of all, the operation on packets in data plane is explained. ICN nodes are the main components of this plane. When an ICN packet, either a request packet or a data packet, arrives at the ICN node, the packet is matched against flow entries in the ICN node's flow table. If packet matching succeeds, the set of action correspond the matched flow entry will be applied on that packet. Otherwise, the ICN node will send *OpenFlow packet-in message* containing packet's information to the controller. Then, the controller will take the actions to respond to that packet according to its coding algorithm. The action set is sent to the ICN node via OpenFlow message. The ICN node will follow the action set from the controller. The sequences of operation while there is no packet matching with flow entry in the ICN's flow table are illustrated in Figure 3.13.

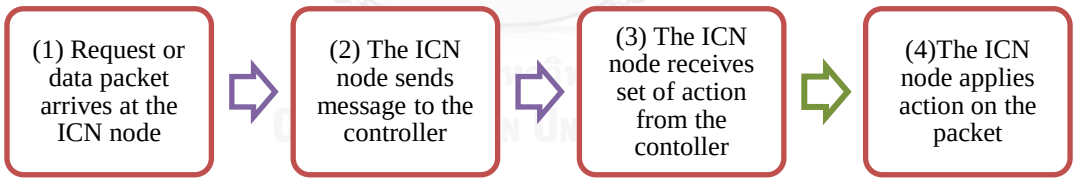

Figure 3. 13: Sequences of operation on packets at the ICN node while there is no packet matching

3.2.4.2 Operation in server

When a first request packet reaches the content server; the server, a PC installed with open vswitch software, sends packet's information to the controller via *OpenFlow packet-in message*. The controller will modify and install flow entries in the content server's flow table by attaching the set of actions on the request packet such as firstly swapping the request packet's source MAC address to destination MAC

address and vice versa, that packet will become the data packet as being explained in section 3.2.2, then forwarding the packet to its incoming port for every incoming packet. Those actions are sent to the content server by using OpenFlow message. The server adopts the actions from the controller. After the flow entry is installed, next incoming request packets will be matched in server's flow table and the set of actions will be applied. The server does not need to send the request packet's information to the controller again later. It acts as the real server by using the request packet itself to be the data packet.

3.2.4.3 Operation in controller

The controller is a component in which the in-network caching and content visibility process coding is implemented. In this part, only some crucial operations are presented. First of all, the operation of identifier attachment in the packet is demonstrated. The ICN node transmits packet's information to the controller via *OpenFlow packet-in message* when it cannot find packet matching in its flow table. Assuming that, according to network state, the controller decides to direct the packet to a specific ICN node in the network. An identifier is needed to attach in the packet header as being described in the section 3.1.3.2, the controller commands the ICN node to add a UDP destination port number to that packet's header. The UDP destination port number is used as the identifier in our testbed. The set of commands is sent to ICN node via OpenFlow message.

Another important operation is data reply. While the request packet arrives at the ICN node containing required data, the controller helps coordinate the data reply in our experiment because the ICN node does not store the real contents; the request packet is used as the data packet. This is one of the drawbacks of our testbed implementation. The controller instructs the ICN node to swap the request packet's source MAC address to destination MAC address and vice versa, then forward the packet to its incoming port. That request packet now becomes the data packet after MAC address swapping. The identifier might be added to the packet header also.

The last most important operation is content caching process. As being mentioned in the path cooperative in-network caching policy with path content visibility level mechanism and the global cooperative in-network caching policy with global content visibility level mechanism section, the content is cached at the ICN node and caching updating information is stored in controller's lists. However, in our experiment, only the name of content is kept in the controller list and the real content is not cached in any ICN nodes. Once the controller decides to cache the content in any ICN node and when the content arrives at that ICN node, the ICN node will update information to the controller. The controller will then update the name of content in its list.

In conclusion, according to the limitation of our testbed, none of data packets is stored in the ICN node in our testbed. Instead, only the lists of contents of all ICN nodes are recorded in the controller. This causes some slightly differences from our centralized in-network caching policy with content visibility level mechanism as well as the real application. Moreover, this experiment will affect the message rate between control and data plane metric of non-cooperative in-network caching policy with individual content visibility mechanism because the caching process is done in the controller. But, in non-cooperative in-network caching policy with individual content visibility mechanism, the caching is conducted in ICN nodes. Nevertheless, other performance metrics are still valid with this experiment.

3.3 Summary

This chapter explains the detail of the three in-network caching policy with content visibility level mechanisms deployed in our work. Each mechanism consists of an in-network caching policy and a level of content visibility. The in-network caching and content searching operation of each mechanism were presented. It also gives the information of the PC-based OpenFlow testbed implementation. The detail of network component implementation, software used, ICN packet design and other operations were discussed in this chapter.

CHAPTER 4

PERFORMANCE EVALUATION

This chapter intends to evaluate the performance of the ICN over SDN by considering the three centralized in-network caching policy with content visibility level mechanisms as being mentioned in section 3.1. This part also describes the experimental setup in both PC-based OpenFlow testbed and emulation environments. Moreover, performance metrics, necessary parameters and assumption in the experiment are pointed out as well. Finally, results of ICN's performance are discussed.

4.1 Performance Metrics

To evaluate the performance of ICN under SDN support, four performance metrics are taken into account. They are *server hit ratio, average hop count, message rate between control and data plane*, and *bottleneck link traffic*. It is believed that these performance metrics will show both strength and weakness of each proposed mechanism. The smaller values of performance metrics are the better performance obtained.

1) Server hit ratio: It is the ratio of number of requests that hit the server over the total requests of all requesters in the network.

2) Average hop count: It is the average number of ICN nodes that one request packet needs to pass through for fetching the desired data content.

3) Message rate between control and data plane: This refers to total number of communication messages, OpenFlow messages [1], sent in both directions (controller-switches) between control and data plane per time unit.

4) Bottleneck link traffic: It aims to measure the traffic in both directions at the bottleneck links in the network.

4.2 Parameter selection and assumption

In order to reduce the complexity of the experiment, it is necessary to make assumption of many parameters. This investigation focuses on content level. One

packet is referred to one content; the content is not divided into small chunks. Below are parameters and processes assumed and utilized in this work.

4.2.1 Content replacement policy

Content replacement policy is applied when the ICN node's cache is full. One or more contents will be removed from the content storage to give the space for the new content. According to many research papers such as [23], [24], [25], content replacement policies do not affect much to ICN's performance. For example, the random replacement policy gives quite the same performance as the Least recently used (LRU) [24], [25]. In this thesis, random replacement policy is selected. It means that one content will be removed randomly from the cache if the space is needed.

4.2.2 Cache dimensioning, size of contents and numbers of content object

Cache dimensioning also has influence on the network performance. In the design concept of ICN, the cache should operate at line rate [13], so the size of cache should be limited. In previous works, the cache size varies from 500KBytes [26] to 10GBytes [25]. On the other hand, many research papers considered different numbers of content objects in the network. Reference [19], [27] and [28] allowed having 20000 different content objects in the network. [25] took the number of contents equal to 10^8, and 10000 contents was taken by [26]. However, in [29], there were only 900 different contents. If we look at the size of content, [26] fixed the size of content to be 1KBytes, [18] took the size of content from 4 to 13KBytes while [25] and [29] took 10MBytes and 140KBytes respectively. The size of content and cache in [19] were 6.9MBytes and 2GBytes respectively. The size of cache, content and number of contents varied according to the application, i.e. video, files, etc. Anyway, in our research, the small network is considered, so the following values were decided to use:

- Number of content : 1000 contents
- Content's size : 1024 byte

- Cache size : 50, 75 and 100 contents

The size of content cannot be more than 1500 byte because it is the maximum transmission unit at Ethernet port. Moreover, the cache sizes in this study are 50, 75 and 100 contents. This work plans to limit the cache size not to exceed the total number of contents in one class.

4.2.3 Content popularity and request pattern

Requesters request contents based on content popularity. Requester requests the most popular contents quite often while it requests fewer unpopular contents. In this work, all requesters have the same request pattern. It follows the Zipf popularity distribution [19], [25]. The Zipf distribution divides contents into N classes and the popularity of class k where k $= \{1,2,3\ldots,\mathrm{N}\}$ is defined by

$$p_k = \frac{1/k^{\alpha}}{\sum_{n=1}^{N} 1/n^{\alpha}} \quad (1)$$

where α is Zipf exponent. It characterizes the skewness of popularity [19]. It means that if α is large, there are few contents' classes with high popularity; if α is small, there are many contents' classes with similar popularity but not so high (Figure 4.1). On the other hand, assuming that the total request rate is λ_t, so the request rate for each class is $\lambda_k = p_k \lambda_t$. The time interval between two requests of a requester is fixed. In our research, the total contents are divided to 10 classes, each class has $n = 1000/10 = 100$ contents. The value of exponent is $\alpha = 2$. Also, contents in the same class have the same probability to be requested.

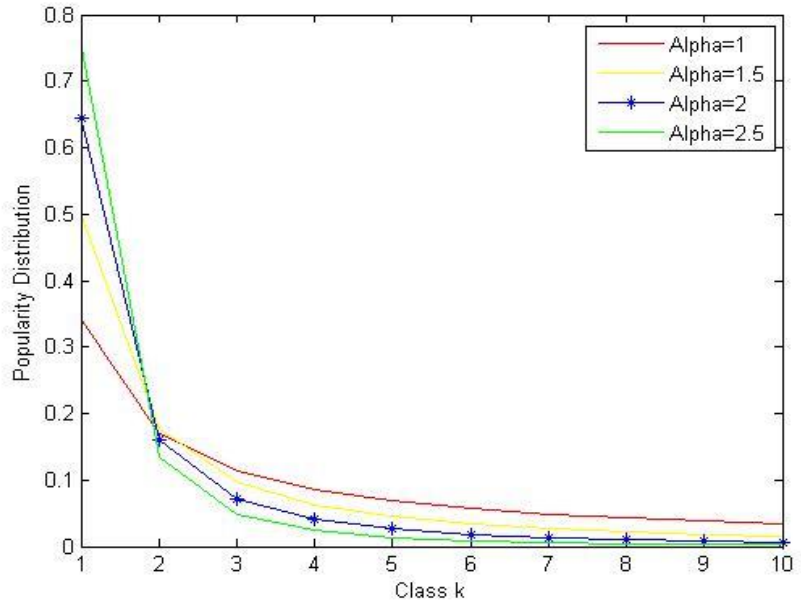

Figure 4. 1: Zipf popularity distribution of ten classes of content with different values of α

4.3 Experimental setup

4.3.1 General setup

This work performs the experiments in two different experimental environments. They are PC-based OpenFlow testbed as being described in section 3.2 and emulation by using Mininet [30]. The parameters used in these two environments are identical as shown in Table 4.1 that is the summary from section 4.2. Moreover, the experiments are conducted over two network topologies, the cascade and the three-level binary tree topology. The cascade network topology consists of one network controller, one content server, two content requesters and six ICN nodes as depicted in Figure 4. 2. The second topology is three-level binary tree topology. It comprises four content requesters, one content server, one network controller and seven ICN nodes as shown in Figure 4.3.

Table 4. 1: Parameters for experiment

Parameters	Value
Number of total contents	1000
Number of content classes	10
Number of contents per class	100
Request rate for each requester	$\lambda = 5/\sec$
Content size	1Kbyte
Total requests for each requester	5000
Zipf exponent	$\alpha = 2$

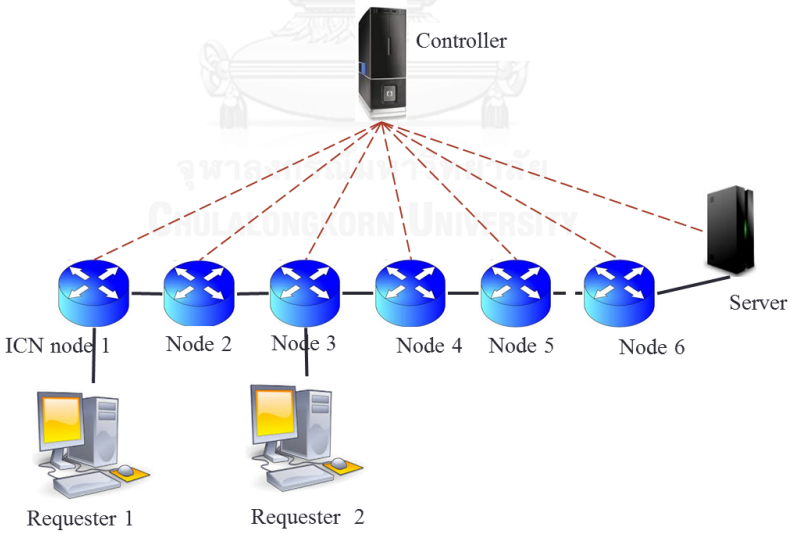

Figure 4. 2: Cascade topology

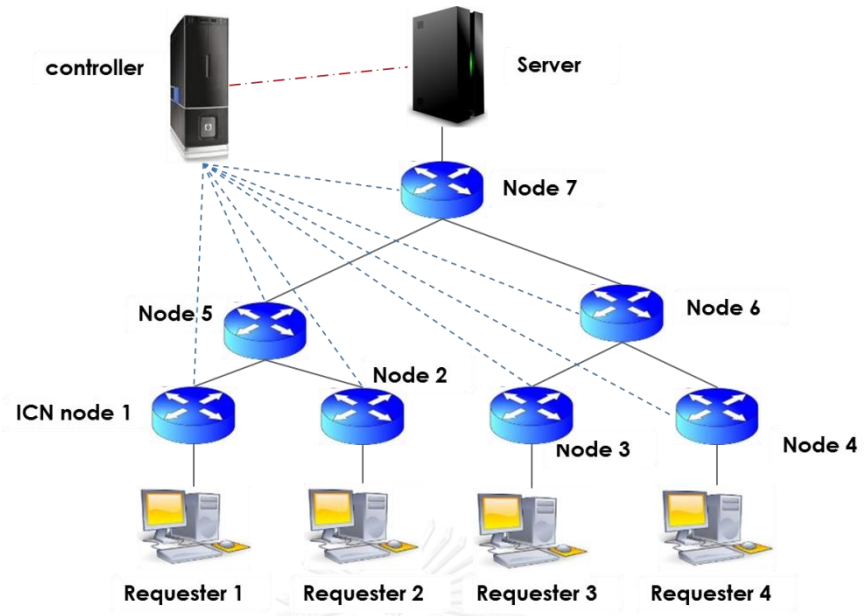

Figure 4. 3: Three-level binary tree topology

4.3.2 Implementing the three mechanisms in cascade and tree topology

How to implement the three mechanisms as being mentioned in section 3.1 in the cascade and tree topology is described here.

4.3.2.1 Non-cooperative in-network caching policy with individual content visibility mechanism

As described in section 3.1.1, the content visibility level of this mechanism is individual; it means that each ICN node searches for the desired data content in its own storage only. The request packet is forwarded to the next ICN node along the shortest path to the server if the current ICN node does not have the desired content. There are two shortest paths from requesters to the server in cascade topology as depicted in Figure 4.2. The *cascade_path_1* is the path from the ICN node 1 to the server; *cascade_path_2* is the path from the ICN node 3 to the server. Also, ICN nodes copy all data contents passing through them.

The same concept as aforementioned is applied in tree topology (Figure 4.3). There are four shortest paths from requesters to the server. The *tree_path_1* consists of the ICN node 1, node 5, node 7 and the server. The *tree_path_2* consists of the

ICN node 2, node 5, node 7 and the server. The *tree_path_3* consists of the ICN node 3, node 6, node 7 and the server. The *tree_path_4* consists of the ICN node 4, node 6, node 7 and the server.

4.3.2.2 Path cooperative in-network caching policy with path content visibility level mechanism

In cascade topology, the controller can look for the desired data in ICN nodes along the *cascade_path_1* when it gets the request from Requester 1. However, the controller can only search for the required data content in the *cascade_path_2* if the request is from the Requester 2. According to section 4.2.3, the request pattern of each requester follows the Zipf popularity with value of exponent $\alpha = 2$ and the contents are divided into 10 classes. It means around 64% and 16% of total requests are contents of class 1 and class 2 respectively. The other 20% of total requests are contents of class 3 to class 10. Based on this content popularity proportion, if the Requester 2 requests contents, the contents are cached in the *cascade_path_2* as follow:

- Contents of class 1 are stored in the ICN node 3 while the ICN node 3's cache is not full. But, when the cache of ICN node 3 is full, contents are cached in the ICN node 4 (Figure 4.2). When both ICN nodes' caches are full, one among those two ICN nodes is randomly chosen to keep the content; the old content is removed and replaced by the new content.
- Contents of class 2 are cached in the ICN node 4 together with contents of class 1. If the cache of the ICN node 4 is full, the old content is randomly removed to give space for the new content.
- Contents of other classes are stored in the ICN node 5 and ICN node 6

If the Requester 1 requests data contents, contents are cached in the *cascade_path_1* as follow:

- Contents of class 1 are cached in the ICN node 1 when the ICN node 1's cache is not full. The ICN node 2 stores the contents when the cache of ICN node 1 is full. When both ICN nodes' caches are full, one among those two

ICN nodes is randomly selected to cache the content; the old content is removed and replaced by the new content.

- Contents of class 2 are stored in the ICN node 2. If the cache of the ICN node 2 is full, the old content is randomly removed to give space for the new content.
- Contents of other classed are cached in ICN node 5 and ICN node 6

In tree topology, the controller can search for desired data contents in ICN nodes along the *tree_path_1, tree_path_2, tree_path_3,* and *tree_path_4* if the requests are from the Requester 1, Requester 2, Requester 3, and Requester 4 respectively (Figure 4.3). In this network topology, the request pattern of each requester is same as which in the cascade topology. If the contents are requested by the Requester 1, they are stored in ICN nodes of the *tree_path_1* as follow:

- Contents of class 1 are cached in the ICN node 1 if the ICN node 1's cache is not full yet. When the cache of ICN node 1 is full, the contents are stored in the ICN node 5. On the other hand; if both ICN nodes' caches are full, one of those two ICN nodes is randomly selected to store the content.
- Contents of class 2 are cached in the ICN node 5 while the space of that ICN node's cache is available. When the cache of the ICN node 5 is full, the old content is randomly removed and replaced by the new content.
- Contents of the other classes are stored in the ICN node 7 only. One content is randomly removed if the space is needed for the new content.

If the contents are requested by the Requester 2, they are stored in ICN nodes of the *tree_path_2* as follow:

- Contents of class 1 are cached in the ICN node 2 if the ICN node 2's cache is not full yet. When the cache of ICN node 2 is full, the contents are stored in the ICN node 5. On the other hand; if both ICN nodes' caches are full, one of those two ICN nodes is randomly selected to store the content.
- Contents of class 2 are cached in the ICN node 5 while the space of that ICN node's cache is available. When the cache of the ICN node 5 is full, the old content is randomly removed and replaced by the new content.

- Contents of the other classes are stored in the ICN node 7 only. One content is randomly removed if the space is needed for the new content.

If the contents are requested by the Requester 3, they are stored in ICN nodes of the *tree_path_3* as follow:

- Contents of class 1 are cached in the ICN node 3 if the ICN node 3's cache is not full yet. When the cache of ICN node 3 is full, the contents are stored in the ICN node 6. On the other hand; if both ICN nodes' caches are full, one of those two ICN nodes is randomly selected to store the content.
- Contents of class 2 are cached in the ICN node 6 while the space of that ICN node's cache is available. When the cache of the ICN node 6 is full, the old content is randomly removed and replaced by the new content.
- Contents of the other classes are stored in the ICN node 7 only. One content is randomly removed if the space is needed for the new content.

If the contents are requested by the Requester 4, they are stored in ICN nodes of the *tree_path_4* as follow:

- Contents of class 1 are cached in the ICN node 4 if the ICN node 4's cache is not full yet. When the cache of ICN node 4 is full, the contents are stored in the ICN node 6. On the other hand; if both ICN nodes' caches are full, one of those two ICN nodes is randomly selected to store the content.
- Contents of class 2 are cached in the ICN node 6 while the space of that ICN node's cache is available. When the cache of the ICN node 6 is full, the old content is randomly removed and replaced by the new content.
- Contents of the other classes are stored in the ICN node 7 only. One content is randomly removed if the space is needed for the new content.

Note that there is no duplicated content stored in ICN nodes along a path in both cascade and tree topology. All contents in the path are different from each other.

4.3.2.3 Global cooperative in-network caching policy with global content visibility level mechanism

According to global content visibility level of this mechanism in section 3.1.3.1, the locations of all data contents are seen by the controller. If the desired data packet is stored in any in-network ICN node, the controller will forward the request packet to be served by the in-network ICN node storing such that content first in both cascade and tree topology. In case that there is no required data content storing in ICN nodes, the request will be forwarded to the server.

In cascade topology, based on the global in-network caching policy and the content popularity, contents are cached as follow if Requester 1 or Requester 2 requests such contents:

- Every content packet of class 1 is duplicated; one content packet is stored in the ICN node 3 and another is stored in the ICN node 1 if the caches of those ICN nodes are not full. When the caches of those ICN nodes are full, one content packet is stored in the ICN node 4 and another is stored in the ICN node 2. In case that the caches of those four ICN nodes are full, a pair of ICN nodes (ICN node 1, 3 and ICN node 2, 4) is selected to store the same content by replacing the old content randomly.
- Every content packet of class 2 is also duplicated; one content packet is stored in the ICN node 4 and another is stored in the ICN node 2 if the caches of those ICN nodes are not full. When the caches of those ICN nodes are full the old contents are randomly replaced by the new contents.
- Contents of class 3 to class 10 are stored in the ICN node 5 and the ICN node 6. When the cache of ICN node 5 is not full, contents are stored in ICN node 5. If the cache of ICN 5 is full, the contents are stored in the ICN node 6. In case that both caches of those ICN nodes are full, one ICN node is randomly selected to store the new content by removing the old content.

In tree topology, when the Requester 1 or the Requester 2 or the Requester 3 or the Requester 4 requests the contents, and such those contents are from the server, contents are cached in the ICN nodes in the same manner as which of tree

topology in the path cooperative in-network caching policy with content visibility level mechanism (section 4.3.2.2, tree topology). On the other hand, when the Requester 1 requests the data contents of class 1 and the request is served by the in-network ICN node 2 or the ICN node 3 or the ICN node 4 (edge nodes), the content is cached in the ICN node 1. When the requests are served by the edge nodes; the contents of class 1 are cached in the ICN node 2 or the ICN node 3 or the ICN node 4 if the Requester 2 or the Requester 3 or the Requester 4 requests such contents respectively. If the Requester 1 or Requester 2 requests contents of class 1 or class 2 and the requests are served by the ICN node 6, the contents are cached in the ICN node 5. . If the Requester 3 or Requester 4 requests contents of class 1 or class 2 and the requests are served by the ICN node 5, the contents are cached in the ICN nod 6. The contents of other classes, i.e. class 3 to class 10, are kept in the same ICN nodes after such those ICN nodes reply the data back to the requesters.

4.4 PC-based OpenFlow testbed experiment

This part discusses performance evaluation of the three mechanisms in the PC-based OpenFlow testbed. The experiment is conducted based on the experimental setup in Table 4.1. In this study, the experiment is repeated for four times. The results shown in this part are the average values of the four time experimental results. The codes corresponding to the three mechanisms are implemented in the controller.

The non-cooperative in-network caching policy with individual content visibility mechanism, path cooperative in-network caching policy with path content visibility level mechanism and global cooperative in-network caching policy with global content visibility level mechanism are called in short as non-cooperation, path cooperation and global cooperation mechanism respectively in this chapter.

4.4.1 Results from cascade topology

For the cascade network topology (Figure 4.2), only three performance metrics are taken into account. Because of the simplicity of the network topology, the bottleneck link traffic metric is not included.

4.4.1.1 Server hit ratio

After conducting the experiment, the result of server hit ratio was obtained as depicted in Figure 4.4. The global cooperation mechanism gives better performance compared to others for every cache size. When the cache size equals to 50 contents, more than half of requests were served by the server in the non-cooperation mechanism. However, in the path and the global cooperation mechanism, only 37% and 27% of requests reached the server respectively. Usually, when the ICN's cache size is large, the server hit ratio of all the three mechanisms decreases. This is because while the cache is large, more contents can be copied in the caches of the ICN nodes. Most of requests could be served by the in-network ICN nodes.

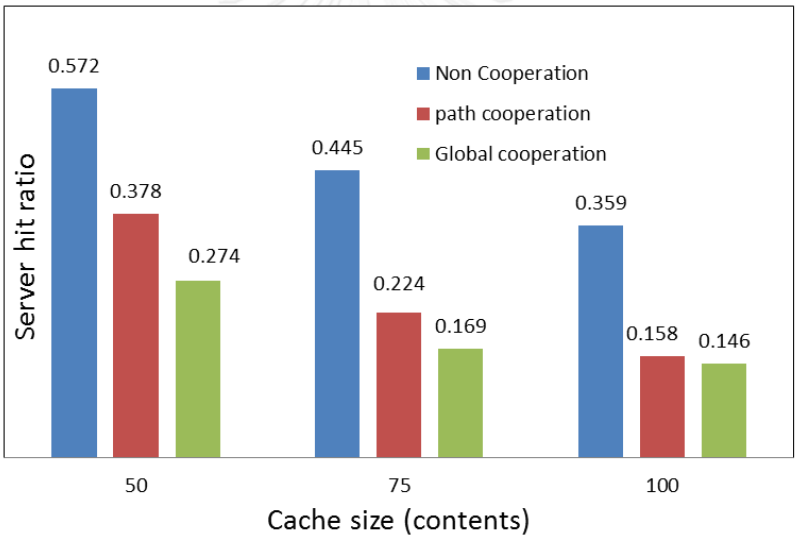

Figure 4. 4: Server hit ratio in cascade topology

4.4.1.2 Average hop count

Figure 4.5 shows the average hop count of the three mechanisms. Again, the global cooperation mechanism outperforms others. To fetch the data content, a request packet travelled across fewer ICN nodes than other mechanisms. In the global cooperation mechanism, the controller can search for the location of content globally (global content visibility level), so it can direct the request packet to be served by the nearest ICN node as possible. This reduced the average hop count.

The non-cooperation mechanism is the worst case in terms of both server hit ratio and average hop count. Not surprisingly; the bigger the cache size is the better performance obtained for all mechanisms.

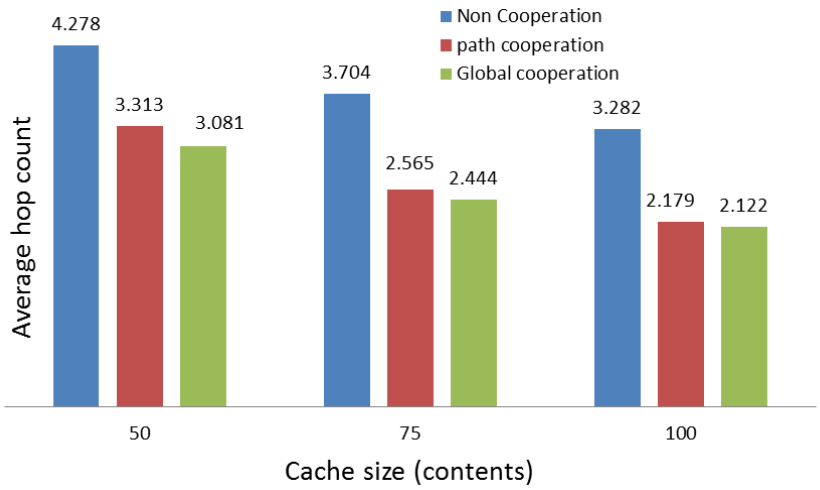

Figure 4. 5: Average hop count in cascade topology

4.4.1.3 Message rate between control and data plane

Figure 4.6 illustrates the message rates between control and data plane. Note that these message rates are slightly different from the definition of the message rates in section 4.1 because the message rates shown in the Figure 4.6 are not the pure OpenFlow messages. They are the TCP messages. In the PC-based OpenFlow testbed experiment, communication messages between control and data plane are taken place through TCP connection, so the pure OpenFlow messages are encapsulated into TCP messages; some overheads such as acknowledgement are included. Based on the PC-based OpenFlow testbed implementation, the in-network caching and content searching processes of the non-cooperation mechanism could not be done at the ICN nodes as being mentioned in section 3.1.1. Instead, these processes were conducted in the controller. Thus, the message rate between control and data plane in the non-cooperation mechanism is not valid. Consequently, only

the message rate in the path and the global cooperation mechanism were discussed in this study. However, in the non-cooperation mechanism, the message rate must be lower than other mechanisms, according to the description in section 3.1.1. For every cache size, the message rates in the path cooperation mechanism and in the global cooperation mechanism are similar.

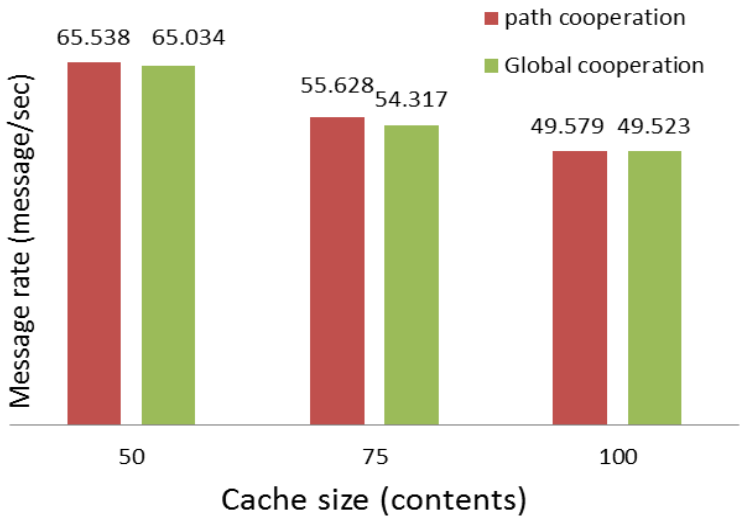

Figure 4. 6: Message rate in cascade topology

This is because the cascade network topology is simple. According to the path and the global cooperation mechanism, most of communication messages between control and data plane take place when the ICN nodes send the request or the data packet information to the controller for performing content searching or content caching decision. The main factor to differentiate the message rate of those two mechanisms is content caching or in-network caching process. For the cascade topology in this work, the global cooperation mechanism did not copy the same data content to store in various ICN nodes. The topology is small and simple, so it is easy to cache the content near to all requesters without keeping the replica in many ICN nodes. It reduces the message rate of the global cooperation mechanism.

4.4.2 Results from tree topology

This part presents the experimental results of the three mechanisms in three-level binary tree topology as depicted in Figure 4.3. All the performance metrics are considered in this topology.

4.4.2.1. Server hit ratio

First of all, the server hit ratio of the three mechanisms is shown in Figure 4.7. Similar to the results from the cascade network topology, the global cooperation mechanism outperforms others. In case that the cache size is small, i.e. 50 contents; the server hit ratio of the global cooperation mechanism is much smaller than the other two mechanisms. It means that most of the requests are served by in-network ICN nodes. More than half of the total requests are served by the server in the non-cooperation and the path cooperation mechanism. But, when the cache size increases, the server hit ratio of the path and the global cooperation mechanism converge to be similar.

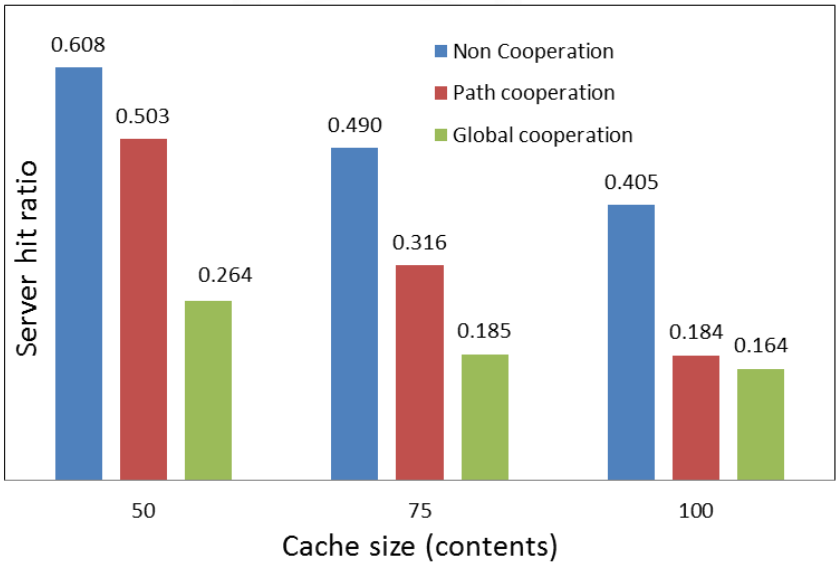

Figure 4. 7: Server hit ratio in tree topology

4.4.2.2 Average hop count

No matter what kind of network topology, the average hop count of the non-cooperation mechanism is still higher than others. The request packet needs to travel across many ICN nodes to get the desired data. Interestingly, the average hop count of the path and the global cooperation mechanism are almost the same for every cache size as depicted in Figure 4.8. In this case, the network became larger, so the request packets in the global cooperation mechanism could travel across many ICN nodes in the network to fetch the data content. But, the caching policy in this mechanism tried to cache the popular contents as close as possible to requesters; it could reduce the hop count. These factors might cause the similarity of average hop count between the path cooperation and the global cooperation mechanism.

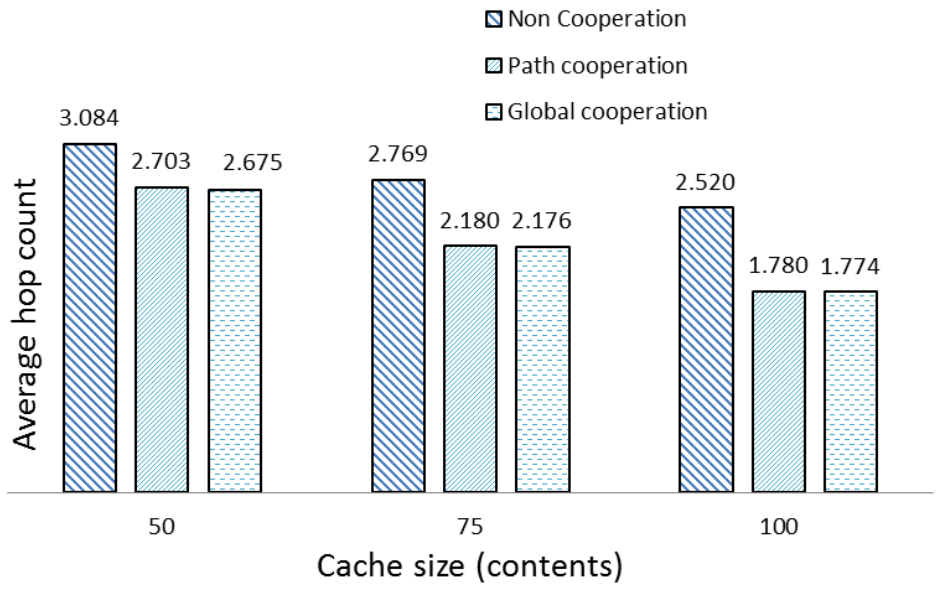

Figure 4. 8: Average hop count in tree topology

4.4.2.3 Message rate between control and data plane

Unlike the result in the cascade topology, the message rates between control and data plane of the path cooperation is lower than those of the global cooperation mechanism in tree topology (Figure 4.9). This is because the request packet might travel across many ICN nodes to fetch the content, hence the communication message between the planes increased. While the network topology

becomes larger and more complex, the global cooperation mechanism shows the disadvantage in terms of the message rate between the planes.

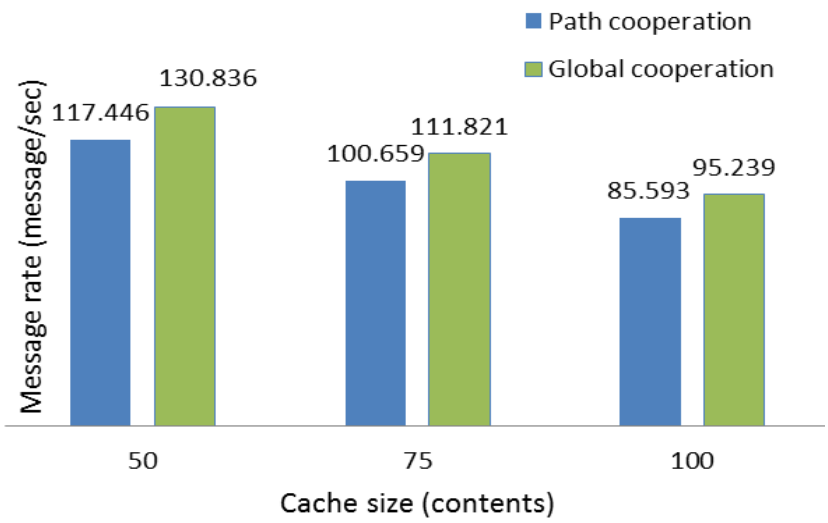

Figure 4. 9: Message rate between control and data plane

4.4.2.4 Bottleneck link traffic

The bottleneck link here refers to the network link that can be caused traffic congestion easily because of high traffic flow. The bottleneck link of the tree topology in this work is the link between the ICN node 5 and the ICN node 7 as depicted in Figure 4.3. The result of bottleneck link traffic is shown in Figure 4.10. The bottleneck link traffic of the non-cooperation mechanism is higher than others. This is because the in-network caching and the content searching process in this mechanism are taken place in the individual ICN node without the coordination from the controller. Hence, the same content might be stored in several ICN nodes' caches. This reduces the diversity of contents in the network. In consequence, many requests are forwarded to fetch the data from the server. On the other hand, when cache size is small, the bottleneck link traffic of the global cooperation is higher than which of the path cooperation mechanism because the requests might be forwarded to and be served at other ICN nodes in another side of the network. Remember that in the global cooperation mechanism, the controller can search for the location of

required data globally. On the contrary, the bottleneck link traffics of the path and the global cooperation mechanism are almost same when cache size increases.

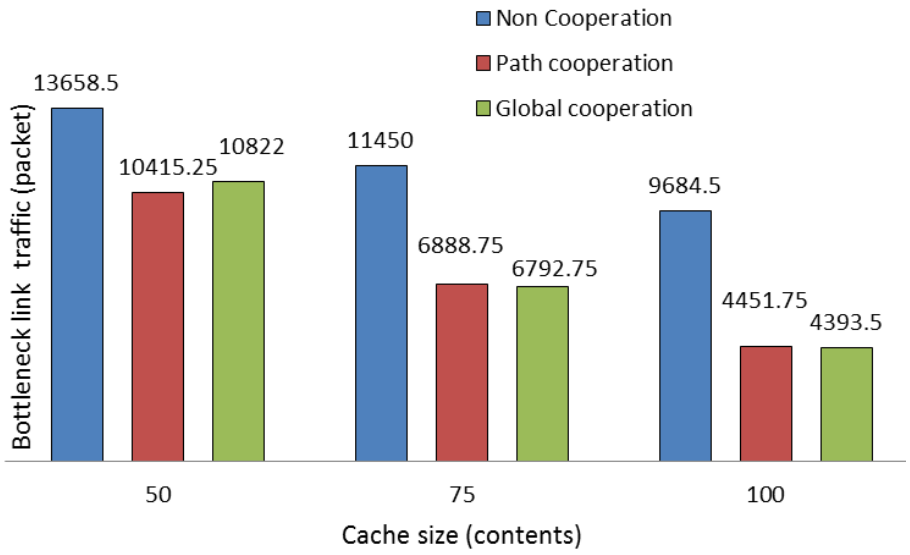

Figure 4. 10: Bottleneck link traffic

4.5 Emulation experiment

In this experiment; all parameters, operations on packets and tagging method are identical to those of the PC-based OpenFlow testbed experimental environment. But, this experiment used virtual network topology. It means that the whole network topology is run in only one computer. Mininet is chosen as the network emulator for the experiment. The ICN nodes and the server in the data plane are virtual nodes installed with open vswitch version 1.9. Requesters are also virtual nodes. These components are created by using Mininet VM image 2.1.0 run in a VirtualBox [31] in Core-i5-1.8 GHz clock speed Sony machine. Then, the POX controller [21], [32] is run remotely in the same VirtualBox. The VirtualBox is set to have 2GBytes of RAM in the experiment.

4.5.1 Results from cascade topology

After conducting the experiment, the results of performance metrics obtained are almost the same as those obtained from the PC-based OpenFlow testbed experiment in the cascade network topology, except the message rate between control and data plane. Figure 4.11, Figure 4.12 and Figure 4.13 demonstrate the results of server hit ratio, average hop count, and message rate between control and data plane respectively. Figure 4.13 shows that the message rates in the cascade topology of this experiment are almost equal to half the message rates obtained in the PC-based OpenFlow testbed experiment. This is because, in this experiment, the virtual network topology was utilized and the experiment was conducted in a virtual box, so the communication message between the controller and the ICN nodes are pure OpenFlow messages. But, in the PC-based OpenFlow testbed, the real computers were utilized as the network components. The connection between ICN nodes and the controller are via TCP connection, so the higher message rate of that experiment was caused by the higher TCP message overheads required. However, the number of pure OpenFlow messages between control and data plane in the PC-based OpenFlow testbed experiment and the emulation experiment must be same.

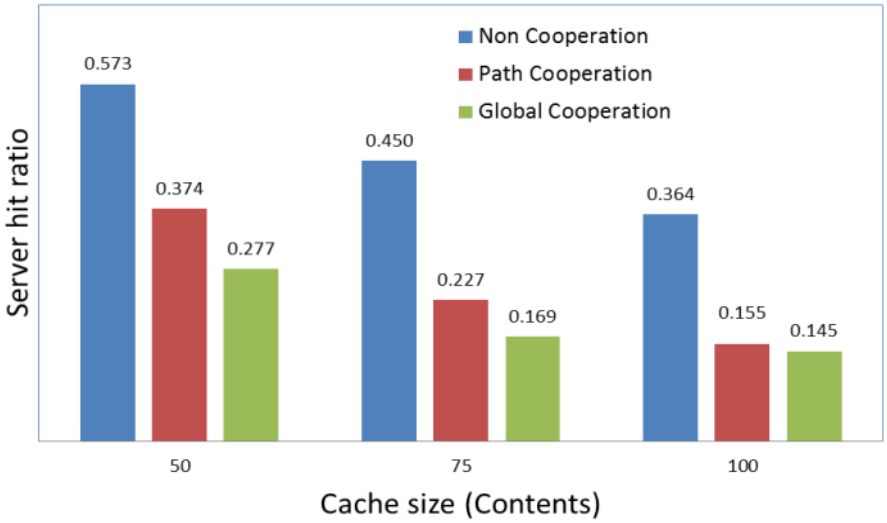

Figure 4. 11: Server hit ratio in cascade topology

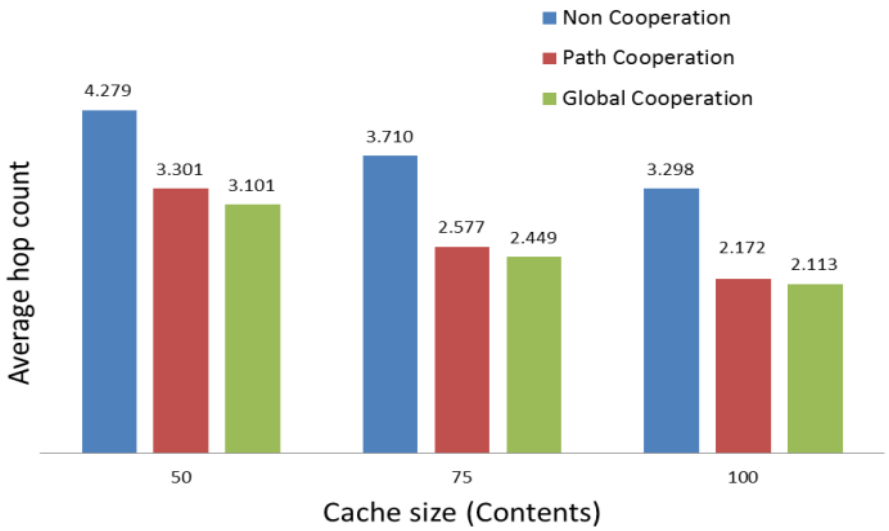

Figure 4. 12: Average hop count in cascade topology

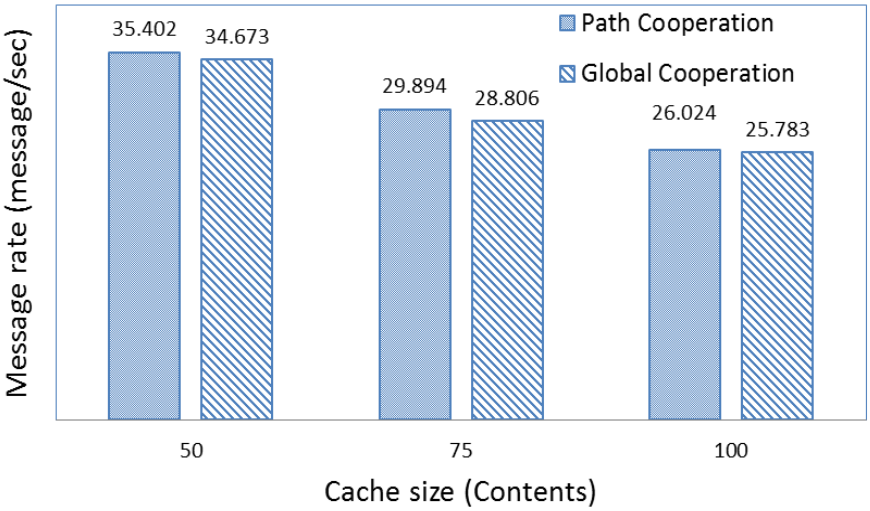

Figure 4. 13: Message rate in cascade topology

4.5.2 Results from tree topology

Again, in the tree topology, the results of performance metrics obtained from the PC-based OpenFlow testbed and from the emulation are almost identical, except the message rate between control and data plane. The results of server hit ratio, average hop count, bottleneck link traffic are illustrated in Figure 4.14, Figure 4.15 and Figure 4.16 respectively.

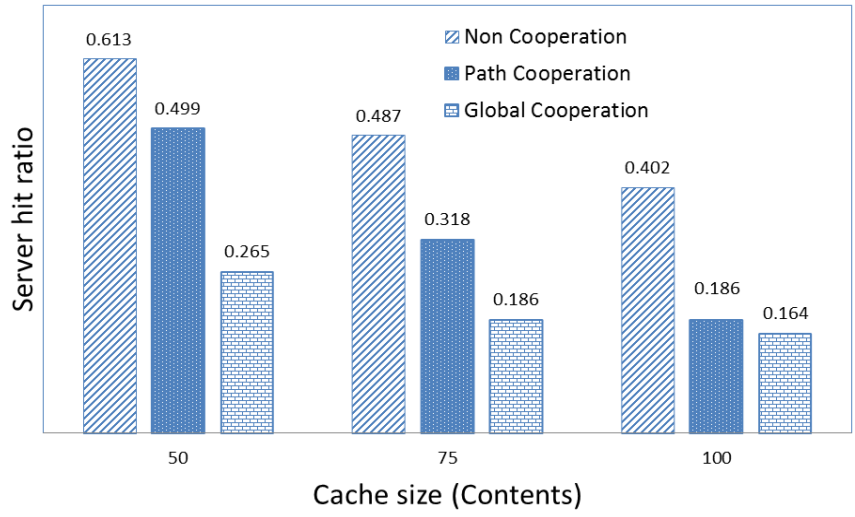

Figure 4. 14: Sever hit ratio in tree topology

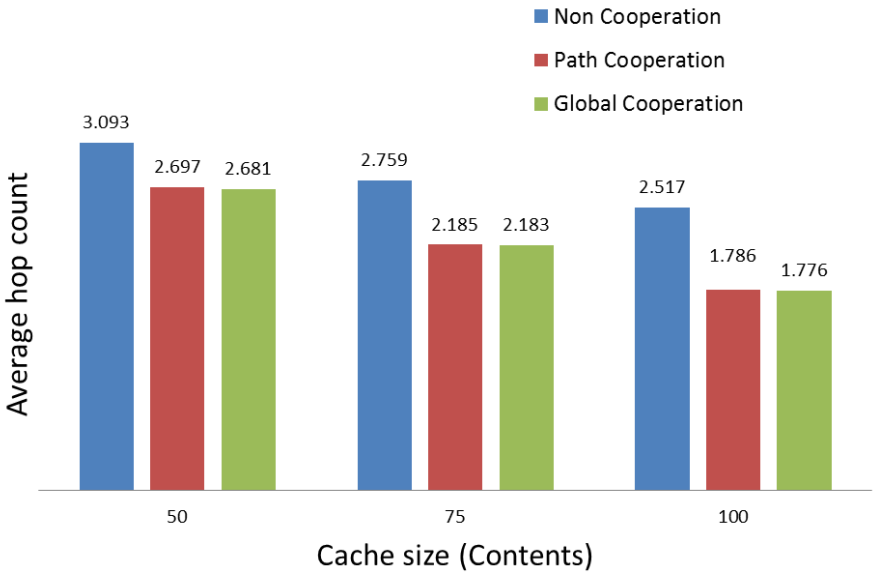

Figure 4. 15: Average hop count in tree topology

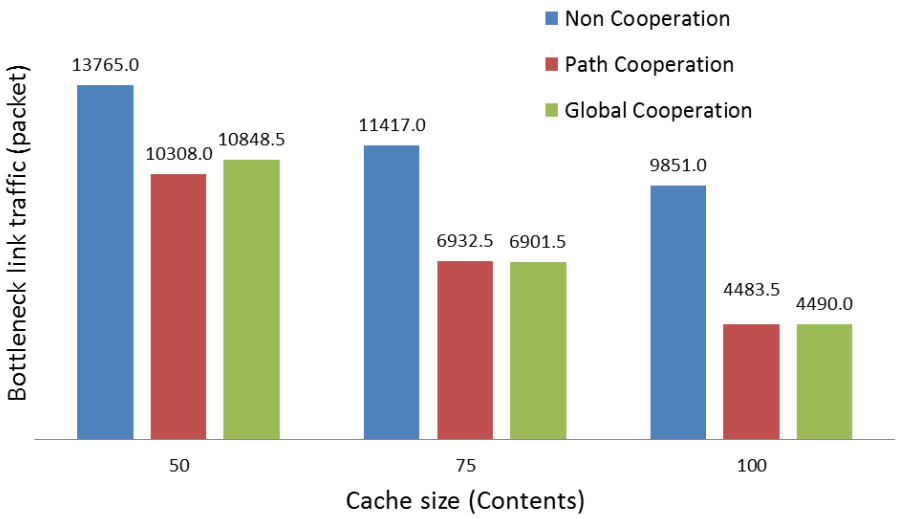

Figure 4. 16: Bottleneck link traffic in tree topology

The message rate between control and data plane is depicted in Figure 4.17. Like the result in the cascade topology, the message rates in the tree topology in the emulation are almost equal to half the message rates obtained in the PC-based OpenFlow testbed experiment. Again, the number of pure OpenFlow messages between control and data plane in these two experiments must be same.

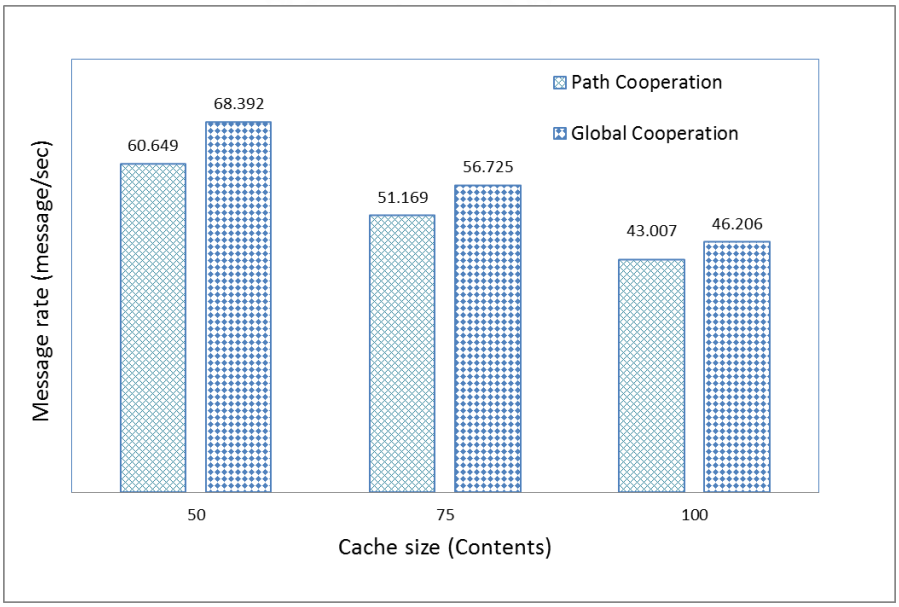

Figure 4. 17: Message rate in tree topology

4.6 Conclusion

This chapter shows the network performance results of the three mechanisms in both PC-based OpenFlow testbed and emulation environment. The parameter selection and assumption, and experimental setup are also described in this chapter. There are four performance metrics taken into account in this work including *server hit ratio, average hop count, message rate between control and data plane*, and *bottleneck link traffic*. Also, two different network topologies, the cascade and the three-level binary tree topology are tested. Each result of performance metric is also discussed in this chapter.

CHAPTER 5

CONCLUSION

The main objectives of this dissertation are to evaluate the performance of information centric network (ICN) under the support of software defined networking (SDN) while three different centralized in-network caching policy with content visibility level mechanisms were deployed. Those three mechanisms consist of non-cooperative in-network caching policy with individual content visibility mechanism or in short “non-cooperation mechanism”, path cooperative in-network caching policy with path content visibility level mechanism or in short “path cooperation mechanism” and global cooperative in-network caching policy with global content visibility level mechanism or in short “global cooperation mechanism”. This work also studies the effect of the network topology on the performance. To realize this work, the detail of sequences of operation for each mechanism and how to implement the PC-based OpenFlow testbed for the experiment were pointed in **Chapter 3**. The experiment and result discussion were in **Chapter 4**.

Performance results from the PC-based OpenFlow testbed and the emulation experiment are almost identical except *message rate between control and data plane*. This is because the message types of these two experiments are different. In the PC-based OpenFlow testbed experiment, the communication messages are TCP messages, the pure OpenFlow messages are encapsulated into TCP messages and some overheads are added. In the emulation experiment, the messages are pure OpenFlow messages. However, the number of pure OpenFlow messages in both experiments must be the same. The emulation by Mininet is recommended to use for the experiment because it consumes less resource and gives almost same results as those in PC-based OpenFlow testbed.

Each mechanism has its own advantages and drawbacks. The result indicates that network topology has significant effects on the performance of ICN over SDN. When the network is small and simple, the global cooperative in-network caching policy with global content visibility level mechanism outperforms others. But, when

network topology becomes larger and complex, this mechanism illustrates some disadvantages in terms of *message rate between control and data plane*, and the *bottleneck link traffic*. The non-cooperative in-network caching policy with individual content visibility mechanism shows the worst performance in almost every performance metrics and network topologies. However, according the mechanism description, the non-cooperative in-network caching policy with individual content visibility mechanism would generate less *message rate* compared to the other two mechanisms. Because of the limitation of our testbed design, the *message rate* in this mechanism is not valid; hence the result was not shown.

On the other hand, the selection of the mechanism is actually based on the network administrators' preference and available resources. For example, if the system needs to reduce content requested from the server, the global cooperation mechanism is the best choice. But, the *message rate between control and data plane* may increase. Nevertheless, these results are applicable only with our parameter setting in one autonomous system. This work is just a starting point to investigate effects of centralized in-network caching policies and content visibility level in ICN under SDN support. Also, the in-network caching process of this work is heuristic; there is no optimization technique used in this research yet.

For the future work, the time-related performance metrics should be included and optimization process for the centralized in-network caching should be considered. Moreover, the further study in the future should use the real files such as video or any types of file in the experiment not just only the packets generated from Scapy program as in the current experiment. Larger and more sophisticated network topology should be studied because, in the ICN architecture design, the network topology is arbitrary not the simple topology. The number of data content, size of content, cache size, etc., could be adopted from the real application such as video or file or the mixture of both file types. Most importantly, the large ICN-based network architecture comprises many autonomous systems should be considered.

REFERENCES

[1] OpenFlow switch specification. version 1.0.0 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[2] Software-Defined Networking: The New Norm for Networks [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[3] SDN architecture overview version 1.0 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf

[4] L. Zhe and G. Simon, "Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1-6.

[5] M. Zhongxing, M. Xu, and W. Dan, "Age-based cooperative caching in Information-Centric Networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, 2012, pp. 268-273.

[6] C. Kideok, L. Munyoung, P. Kunwoo, T. T. Kwon, C. Yanghee, and P. Sangheon, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, 2012, pp. 316-321.

[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE,* vol. 50, pp. 26-36, 2012.

[8] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed," *Computer Networks,* vol. 57, pp. 3207-3221, 11/13/ 2013.

[9] N. B. Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri, "An OpenFlow-based testbed for information centric networking," in *Future Network & Mobile Summit (FutureNetw), 2012*, 2012, pp. 1-9.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, *et al.*, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.,* vol. 38, pp. 69-74, 2008.

[11] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti, "Supporting information-centric functionality in software defined networks," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 6645-6650.

[12] A. Ooka, S. Ata, T. Koide, H. Shimonishi, and M. Murata, "OpenFlow-based content-centric networking architecture and router implementation," in *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, 2013, pp. 1-10.

[13] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks,* vol. 57, pp. 3128-3141, 11/13/ 2013.

[14] Y. Kim and I. Yeom, "Performance analysis of in-network caching for content-centric networking," *Computer Networks,* vol. 57, pp. 2465-2482, 9/9/ 2013.

[15] D. Trossen, G. Parisis, B. Gajic, J. Riihijarvi, P. Flegkas, P. Sarolahti, *et al.*, "Architecture Definition, Components Descriptions and Requirements," 2011.

[16] Software Defined Networking (SDN), OpenFlow and Cisco ONE [Online]. Available: http://www.cisco.com/web/TH/assets/docs/seminar/techupdate_20130712_Software_Defined_Networks_SDN_and_OpenFlow.pdf

[17] *Cisco ONE* Available: www.cisco.com/go/one

[18] S. Paul and Z. Fei, "Distributed caching with centralized control," *Computer Communications,* vol. 24, pp. 256-268, 2/1/ 2001.

[19] K. Suksomboon, S. Tarnoi, J. Yusheng, M. Koibuchi, K. Fukuda, S. Abe*, et al.*, "PopCache: Cache more or less based on content popularity for information-centric networking," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, 2013, pp. 236-243.

[20] Open Vswitch [Online]. Available: http://openvswitch.org/

[21] POX [Online]. Available: http://www.noxrepo.org/pox/versionsdownloads/

[22] Scapy [Online]. Available: http://www.secdev.org/projects/scapy/

[23] J. Hongseok, L. Byungjoon, and S. Hoyoung, "On-path caching in information-centric networking," in *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, 2013, pp. 264-267.

[24] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech2011.

[25] G. Rossini and D. Rossi, "A dive into the caching performance of Content Centric Networking," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, 2012, pp. 105-109.

[26] W. Sen, J. Bi, and W. Jianping, "On Performance of Cache Policy in Information-Centric Networking," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, 2012, pp. 1-7.

[27] L. Muscariello, G. Carofiglio, and M. Gallo, "Bandwidth and storage sharing performance in information centric networking," presented at the Proceedings of the ACM SIGCOMM workshop on Information-centric networking, Toronto, Ontario, Canada, 2011.

[28] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Teletraffic Congress (ITC), 2011 23rd International*, 2011, pp. 111-118.

[29] T. Li, N. V. Vorst, R. Rong, and J. Liu, "Simulation studies of OpenFlow-based in-network caching strategies," presented at the Proceedings of the 15th Communications and Networking Simulation Symposium, Orlando, Florida, 2012.

[30] Mininet [Online]. Available: http://mininet.org/

[31] Virtualbox [Online]. Available: https://www.virtualbox.org/

[32] *POX Tutorial*. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki

APPENDIX

Appendix

List of Publications

C. Hel and C. Saivichit, "PC-based OpenFlow testbed implementation for in-network caching of information centric network," *presented at the 6th AUN/SEED-Net Regional Conference on Electrical Engineering 2014*, Kuala Lumpur, Malaysia, 2014.

C. Hel and C. Saivichit, "Performance evaluation of centralized in-network caching and content visibility in information centric network over SDN/OpenFlow," *in International Conference on Electrical Engineering, Computer Science and Informatics 2014 (EECSI 2014)*, Yogyakarta, Indonesia, 2014, pp. 29-33.

C. Hel and C. Saivichit, "Evaluation of software defined-information centric network's performance by considering in-network caching and content searching mechanism," *in International Workshop on Internet Architecture and Applications 2014*, Chiang Mai, Thailand, 2014, pp. 51-56.

VITA

Chanthan Hel was born in 1989 in Takeo, Cambodia. He got his bachelor's degree in Telecommunication Engineering from Institute of Technology of Cambodia in 2012. Currently, he is a master's degree student at Electrical Engineering Department, Chulalongkorn University. His research interests include: Future Internet, Network Protocol, and Telecommunication Networking.