

福岡大学

博士學位論文

THE DUAL CHANNEL IP-TO-NDN TRANSLATION GATEWAY:  
A HOST-TO-NAME SEMANTIC NETWORK PROTOCOL  
TRANSLATION

令和 5 年 3 月

フェリ ファリアント

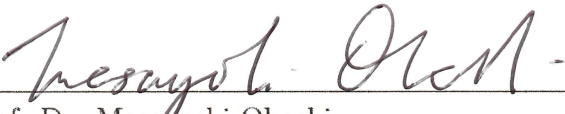
工学研究科 情報・制御システム工学専攻

© 2022 by  
FERI FAHRIANTO  
All rights reserved

Prior to having this page signed by the readers.

Approved by

First Reader

  
Prof. Dr. Masayoshi Ohashi

Second Reader

  
Prof. Dr. Makoto Taromaru

Third Reader

  
Prof. Dr. Hidenori Nakazato

*Anyone who stops learning is old, whether at twenty or eighty. Anyone who keeps learning stays young.*  
— *Henry Ford;*



## **Acknowledgments**

This work was supported by the Ministry of Religious Affairs of the Republic of Indonesia (MORA 5000 Doktor) cooperated with the Indonesian Endowment Fund for Education (LPDP).

**THE DUAL CHANNEL IP-TO-NDN TRANSLATION  
GATEWAY: A HOST-TO-NAME SEMANTIC NETWORK  
PROTOCOL TRANSLATION**

**FERI FAHRIANTO**

Fukuoka University

Graduate School of Engineering

2022

Promoters : Prof. Dr. Masayoshi Ohashi  
Fukuoka University  
Prof. Dr. Makoto Taromaru  
Fukuoka University  
Prof. Dr. Hidenori Nakazato  
Waseda University

## ABSTRACT

The emergence of communication networks has resulted in a system of content-based networks. A novel network design called information-centric networking (ICN) promotes the efficient transfer of content objects, especially through named-data networking (NDN), as one of the most promising candidates. While migration from Internet Protocol (IP) to NDN is inevitable, it is prohibitively expensive to replace all routers instantaneously with NDN routers. As a result, we propose the dual-channel IP-to-NDN translation gateway to address this problem. The gateway promotes the semantics of the IP protocol to be equal to those of NDN at the network layer. The proposed gateway uses two unique IP addresses as channels to distinguish an IP packet as either an interest or data packet without checking the payload of packets. The proposed gateway uses a name-binding mechanism to seamlessly transmit packets between IP and NDN hosts. Using static and dynamic binding schemes, the performance of the dual-channel gateway, especially throughput, is examined. The estimation throughput model is analyzed and evaluated by comparing it with the emulation testbed. The relationship between the hit ratio, processing delay, and throughput is also highlighted. Through numerical evaluation, we show that the throughput estimation model successfully predicts the gateway throughput with an accuracy of about 90% and 85% in static and dynamic prefix-name binding schemes, respectively. Furthermore, a phenomenon called delayed caching that creates a hit ratio decline in the gateway is investigated and analyzed. The suppressing method to prevent the hit ratio decline is also presented.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Related work . . . . .	3
<b>2</b>	<b>Protocol Migration Approaches</b>	<b>6</b>
2.1	Stack modification approach . . . . .	6
2.1.1	Dual-stack approach . . . . .	6
2.1.2	Hybrid approach . . . . .	7
2.2	Encapsulation approach . . . . .	9
2.2.1	Upper-layer encapsulation . . . . .	9
2.2.2	Under-layer encapsulation . . . . .	10
2.3	Translation approach . . . . .	11
2.3.1	Network layer translation . . . . .	12
2.3.2	Transport layer translation . . . . .	12
2.3.3	Application layer translation . . . . .	12
2.4	Migration cost . . . . .	13
<b>3</b>	<b>The Dual-Channel Translation Gateway</b>	<b>16</b>
3.1	Principles . . . . .	17
3.2	Components of dual-channel gateway . . . . .	18
3.3	Packet flows and translation procedure . . . . .	20
3.4	User privacy and security . . . . .	27
3.5	Prefix name preregistration . . . . .	28

3.5.1	Name resolution service . . . . .	28
3.5.2	Registration packet format . . . . .	29
3.5.3	Prefix name registration in scenario 1 . . . . .	30
3.5.4	Prefix name registration in scenario 2 . . . . .	31
<b>4</b>	<b>Throughput Estimation Model</b>	<b>33</b>
4.1	Throughput Analysis . . . . .	33
4.1.1	Static prefix-name binding . . . . .	34
4.1.2	Dynamic prefix-name binding . . . . .	36
<b>5</b>	<b>Throughput Evaluation</b>	<b>41</b>
5.1	Static prefix-name binding . . . . .	42
5.1.1	Hit ratio . . . . .	42
5.1.2	Processing time distribution . . . . .	44
5.1.3	Throughput . . . . .	45
5.2	Dynamic prefix-name binding . . . . .	49
5.2.1	Hit ratio . . . . .	49
5.2.2	Processing time distribution . . . . .	52
5.2.3	Throughput . . . . .	53
<b>6</b>	<b>Delayed Caching</b>	<b>56</b>
6.1	Delayed caching suppressing method . . . . .	58
6.1.1	Burst score . . . . .	59
6.1.2	Cache replacement algorithm . . . . .	62
6.2	Evaluation result . . . . .	63
<b>7</b>	<b>Conclusions and Future work</b>	<b>65</b>
7.1	Conclusion . . . . .	65
7.2	Future work . . . . .	66



## List of Tables

2.1	Symbols related to cost metrics. . . . .	13
2.2	Average migration cost of individual components. . . . .	14
4.1	Definition of delay components. . . . .	34
5.1	Setting values of main parameters in static prefix-name binding scheme. . . . .	42
5.2	Statistical results of delay components. . . . .	44
5.3	Setting value of main parameters in dynamic prefix-name binding scheme. . . . .	50
6.1	Setting values of main parameters . . . . .	63

# List of Figures

1.1	Host-based communication system . . . . .	1
1.2	Named-based communication system . . . . .	2
1.3	Packet mechanism in ICN/NDN router . . . . .	3
2.1	Dual-stack migration approach . . . . .	7
2.2	Hybrid ICN . . . . .	8
2.3	ICN/NDN over IP . . . . .	10
2.4	IP over ICN/NDN . . . . .	11
2.5	Translation gateway . . . . .	11
2.6	IP and NDN co-existence technology. . . . .	14
2.7	Migration cost in IP and NDN co-existence technology. . . . .	15
3.1	Dual-channel IP-to-NDN translation gateway and its prefix-name binding. . . . .	16
3.2	Components of dual-channel IP-to-NDN translation gateway. . . . .	17
3.3	Packet flow translation between IP and NDN protocol in the case of scenario 1. . . . .	20
3.4	Packet flow translation between IP and NDN protocol in the case of scenario 2. . . . .	21
3.5	Translation process in the scenario 1 . . . . .	25
3.6	Translation process in the scenario 2 . . . . .	26
3.7	NRS packet format . . . . .	30



5.1	Comparison of hit ratios of CS when CS capacity is equal to 1% from total number of prefix names in the case of static prefix-name binding scheme. . . . .	42
5.2	Comparison of hit ratios of CS when CS capacity is equal to 10% from total number of prefix names in the case of static prefix-name binding scheme. . . . .	43
5.3	Distribution of processing time components in the case of scenario 1 and static prefix-name binding scheme. . . . .	45
5.4	Distribution of processing time components in the case of scenario 2 and static prefix-name binding scheme. . . . .	45
5.5	Throughput comparison between estimation analysis and emulation testbed when CS size was equal to 1% and 10% from total number of prefix names in the case of scenario 1 using static prefix-name binding scheme. . . . .	47
5.6	Throughput comparison between estimation analysis and emulation testbed when CS size was equal to 1% and 10% from total number of prefix names in the case of scenario 2 using static prefix-name binding scheme. . . . .	48
5.7	Comparison of hit ratio for CS ( $\beta$ ), REG producer ( $\gamma_{p,1}$ ), RREG producer ( $\gamma_{p,2}$ ), REG consumer ( $\gamma_{c,1}$ ), and RREG consumer ( $\gamma_{c,2}$ ) when CS and REG size equal 1 % and 20 %, respectively, of total number of prefix names in the case of dynamic prefix-name binding. . . . .	50
5.8	Distribution of processing time components in the case of scenario 1 and dynamic prefix-name binding scheme. . . . .	51
5.9	Distribution of processing time components in the case of scenario 2 and dynamic prefix-name binding scheme. . . . .	51

5-10	Throughput comparison between analysis and emulation when CS size was 1% of total number of prefix names in the case of scenario 1 and dynamic prefix-name binding scheme. . . . .	54
5-11	Throughput comparison between analysis and emulation when CS size was 1% of total number of prefix names in the case of scenario 2 and dynamic prefix-name binding scheme. . . . .	55
6-1	Cache miss because of delayed caching in different length of $t_{fetch}$ relative to $t_{arrival}$ . . . . .	56
6-2	The hit-ratio ( $\beta$ ) decline in LRU in different value of $t_{fetch}$ relative to $t_{arrival}$ . . . . .	57
6-3	Architecture of BSA based cache replacement algorithm . . . . .	58
6-4	Burst score and Burst score aggregation (BSA) with five sampling period, $t_{fetch}$ , and three unique contents. . . . .	59
6-5	Burst score distribution against skewness parameter ( $\alpha$ ) of Zipf distribution for 1000 unique contents ( $N$ ) . . . . .	61
6-6	Comparison of hit ratio between LRU without delayed caching, proposed method, and LRU . . . . .	64

## List of Abbreviations

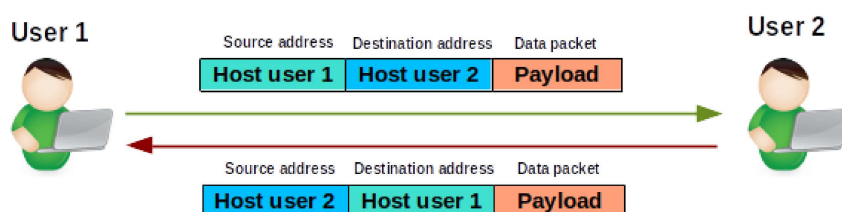
NDN	.....	Named-Data Networking
ICN	.....	Information-Centric Networking
IP	.....	Internet Protocol
CCNx	.....	Content-Centric Networking
FIB	.....	Forwarding Information Base
PIT	.....	Pending Interest Table
CS	.....	Content Store
BSA	.....	Burst Score Aggregation
REG	.....	Register Table
NRS	.....	Name Resolution Service
NLSR	.....	Name Data Link State Protocol
CPA	.....	Content Poisoning Attack
DoS	.....	Denial of Service

# Chapter 1

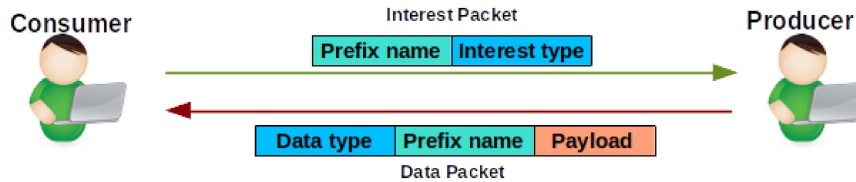
## Introduction

### 1.1 Background

The discovery of Internet Protocol (TCP/IP) has changed the way of communication in the world since its invention in 1974 that initially developed by the Department of Defence (DoD) United States of America (USA). TCP/IP uses a host-centric mechanism for sending and receiving a packet. In the IP packet header, source and destination addresses are stated to establish a connection between the client and the server. This kind of communication scheme is called host-based communication because it utilizes a host address to propagate a packet from one host to another host, as depicted in Figure 1.1. TCP/IP routers forward packets based on the destination IP addresses. They indicate the receiving hosts so that the packets can reach the designated hosts. Currently, TCP/IP is still dominant in wired and wireless communication systems, and connects billions of computers worldwide. Due to an exhaustion of addressing system, evolution from IP version 4 to IP version 6 is required.



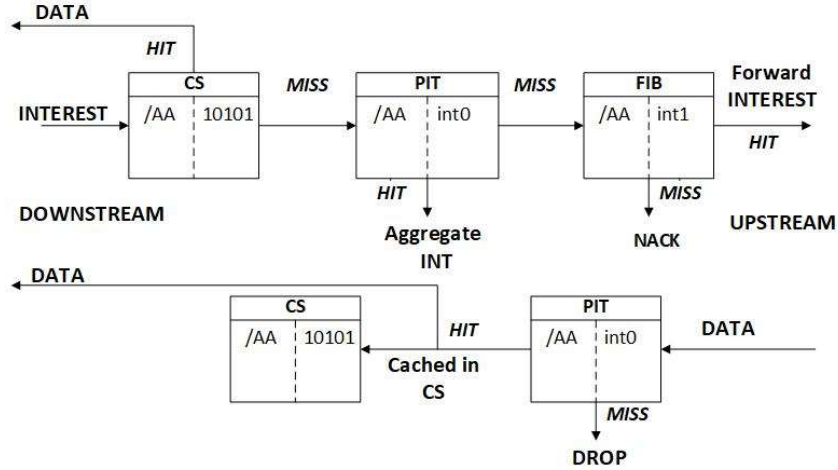
**Figure 1.1:** Host-based communication system



**Figure 1.2:** Named-based communication system

Information-centric networking (ICN) that introduced by Jacobson in [37] revolutionize the paradigm of data exchange in the network by naming the data objects on a network layer. ICN amends the drawbacks of TCP/IP. It enables routers to cache packets for better performance regarding user experience latency. Two distinct packets are required for an ICN data transaction: interest and data packets. To retrieve content from its producer which provides the content, consumer initially sends an interest packet consisting of a content name which is called as prefix name in this document, and the producer responds with a data packet binded with that prefix name as illustrated in Figure 1.2. Moreover, ICN packets are routed based on their prefix name that are stated in interests or data packets. The ICN communication mechanism generally works in a pull-driven manner, in contrast with IP, which may work in a push-based manner.

As ICN technology emerge, named-data networking (NDN) attracts the attention of major communication network vendors. NDN routers have three generic components to forward and process the interest and data packet namely, Pending Interest Table (PIT), Forwarding Information Based (FIB), and Content Store (CS) as seen in Figure 1.3. PIT is used to register prefix names of interest packet and their corresponding incoming interface if the desired content is not found in the CS. FIB is used to route the missed interest packets to other routes trough the interface listed in the table. And finally, CS is used to cache requested data content for future requests. The implementation of NDN technology will be deployed eventually in the worldwide



**Figure 1.3:** Packet mechanism in ICN/NDN router

communication networks. However, co-existence of IP and NDN will be unavoidable throughout the transition period. Lesson learned from IPv.4 to IPv.6 migration, the transition period takes a relatively long time. As a result, a transition technology such as a translation gateway or proxy that enables cross-platform communication between IP and NDN protocols is desired to guarantee a flawless and smooth conversion. Additionally, the transition technology also must consider a cost-effective migration by eliminating the need for numerous network hardware nor software upgrades.

## 1.2 Related work

Many authors have written and recommended about IP and ICN/NDN migration. An extreme suggestion is to redevelop all IP applications with NDN-friendly protocols from zero, which requires an enormous amount of cost and time. VoCCN [38], ACT [46], and DASC [42] are examples of such applications. Even though redesigning all TCP/IP-based applications from scratch is possible, it is impractical from a cost-time perspective. A more reasonable solution is to convert packets from the IP protocol to the NDN protocol by using network protocol migration technology.

A partial migration strategy has already been adopted by Carofiglio et al. in hICN [17] that integrates an ICN/NDN packet inside the IP stack. hICN reuses packet formats deriving the ICN/NDN prefix name from a combination of IP and port headers. It requires minimal software updates and ensures backward compatibility with IP networks. However, it demands a pair of hICN routers to extract the hidden NDN packet from the IP packet as a minimum deployment. Moreover, the hICN limits the length of prefix names, which potentially causes a problem for longer and more varied prefix names. In addition, it also uses the packet flow direction to distinguish IP packets as interest or data packets, which is inefficient for recognizing whether the packet type is interest or data.

Moisennko et al. proposed a TCP/ICN translation proxy pair that can carry TCP traffic over an ICN infrastructure, namely forward proxy and reverse proxy [23]. The proxy approach managed to work correctly, but this approach focuses on the TCP handshaking mechanism, not semantic protocol translation. Similar to Moiseenko, Rafaei et al. in [36] did translation by using a gateway that consists of four components: a gateway daemon for accepting packets, gateway configuration file for naming and configuring packets, traffic handler (TH) for forwarding packets, and command line interface (CLI) for monitoring. However, each IP packet translation process in [36] is based on a full matched-packet filter that requires extra regex computation as overhead. Moreover, this strategy shows potential security breaches such as data privacy violations because it reads the data payload to construct a prefix name and bind a packet.

Luo et al. [34] proposed methods for translating TCP/IP-based packets into NDN packets at three different layers: internet, TCP, and application. The main idea is to name IP packets by using a unique hierarchical naming scheme. By simulating methods using CCNx for FTP servers and clients, the authors showed the feasibility

of multi-level IP/NDN translation and migration. However, the translation is not free from drawbacks. One is the usage of an interest packet to report the causes of the overhead mechanism to the NDN protocol. Another is the enormous size of CS for pre-storing the data. The dual-channel translation gateway focuses on building a translation gateway that obeys the driven-pull semantics in the NDN family protocol with a reusable infrastructure and minimal upgrading overhead.

The rest of the report is organized as follows. The next chapter describes several approaches in network protocol migration technology. Chapter 3 describes the architecture of proposed IP-to-NDN translation gateway like its principles, components, packet flow translations, and binding schemes. In Chapter 4, we introduce the throughput-estimation model of the gateway. Chapter 5 reveals the numerical evaluation results by comparing the analytical calculation with emulation. Furthermore, we discuss a delayed caching phenomenon and its suppressing methods that occurs in the evaluation result of the proposed translation gateway in chapter 6. The final chapter, chapter 7, concludes our contributions and explains the expected future works.



## Chapter 2

# Protocol Migration Approaches

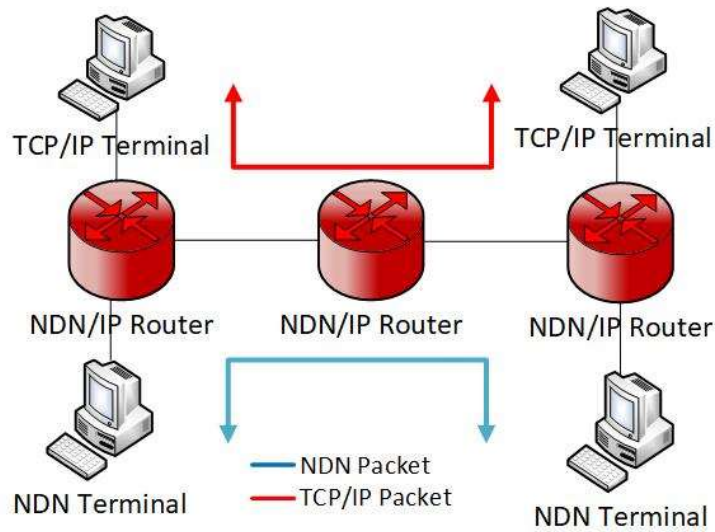
In this chapter, several approaches that are used to migrate from TCP/IP to ICN/NDN are discussed briefly. Some approaches use the translation mechanism, while others use the stack reframing or encapsulation approach in the network protocol migration.

### 2.1 Stack modification approach

Stack restructure in the network protocol is one of the candidates for the migration approach. A particular dual-stack router is a well-known to implement this approach. The dual-stack routers able to distinguish both the TCP/IP header and the NDN header. Currently, stack modification approach has two types of implementation. The first approach uses a dual-stack protocol working in the network while the other uses an hybrid-stack protocol.

#### 2.1.1 Dual-stack approach

The dual-stack approach means that the router can receive and forward the incoming packet independently for both NDN and TCP/IP packet. This proficiency must apply to all routers in the network. The dual-stack approach considers as the most cost burden approach because it replaces all the current routers in the network. In [22], the authors show an implementation of the dual-stack switches to forward NDN packets in the Local Area Network (LAN). Using the MAC addresses, the



**Figure 2-1:** Dual-stack migration approach

authors manage to build the FIB, PIT, and CS tables. By using the dual-stack approach, each packet running in the network uses its header independently. The dual-stack router processes the packets individually regarding their type of NDN or IP, as shown in Figure 2-1. One of the benefits of this approach is that no overhead occurred in the packet header.

### 2.1.2 Hybrid approach

This approach aims for semi dual-stack and partial migration. By modifying the IP header and TCP/UDP header for creating the ICN prefix name, the authors in [17] introduce hybrid ICN or hICN. In the hICN, the ICN prefix name is forced to be fitted and encoded in the IP address header. A regular IP router can recognize an hICN packet as a standard IP packet in the network. However, the hICN router manages to obtain extra information from the source or destination IP header as ICN prefix name.

Hybrid ICN conceals prefix name in IP version 6 (IPv6) addressing scheme. It encodes the prefix name into 128 bits of the IPv6 address. As supplementary, the

hICN uses TCP or UDP header as a suffix name. The concatenation of prefix and suffix creates an encoded ICN prefix name in interest or data packet. The hICN can only use a fixed length of the prefix name instead of a variable length. By putting ICN/NDN name into IP address, hICN enables regular IP routers to propagate a named-centric packet in its network. Thus, partial router upgrading is sufficient for implementing hICN that leads to migration cost reduction.

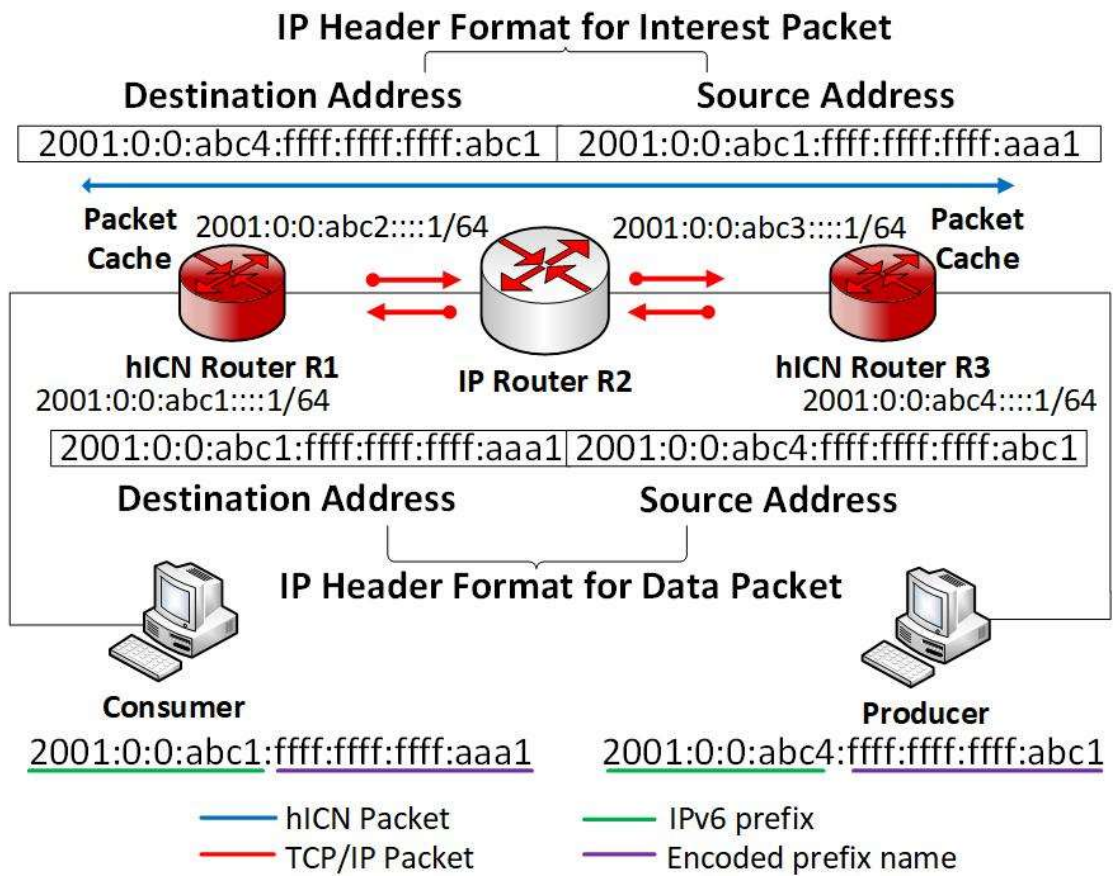


Figure 2·2: Hybrid ICN

Figure 2·2 shows an example where three routers (R1, R2, and R3) exist in the network. Two routers, R1 and R3, are hICN routers, and one router, R2, is a regular IP router. Four unicast IPv6 network addresses set to 2001:0:0:abc1::0/64,

2001:0:0:abc2::0/64, 2001:0:0:abc3::0/64, and 2001:0:0:abc4::0/64 respectively. Since the maximum length of IPv6 addresses are 128 bits, the maximum encoded ICN prefix name is 64 bits. An encoded prefix name ffff:fff:fff:abc1 is created at the consumer. A consumer sends an interest in the 2001:0:0: abc1/64 network by writing 2001:0:0:abc4:fff:fff:fff:abc1 in destination segment of IPv6 header. The interest packet is propagated from R1 to R3 through R2. The intermediate router, R2, processes this interest packet as a regular IP packet and forwards it to network 2001:0:0:abc4::0/64 through R3. R3 forwards this packet to the producer. After receiving the interest, the producer replies with a data packet by rewriting 2001:0:0:abc1:fff:fff:fff:aaa1 in destination segment and 2001:0:0: abc4:fff:fff:fff:abc1 in source segment of IPv6 header. Afterward, the data packet is cached in packet cache located in hICN router R1 and R3.

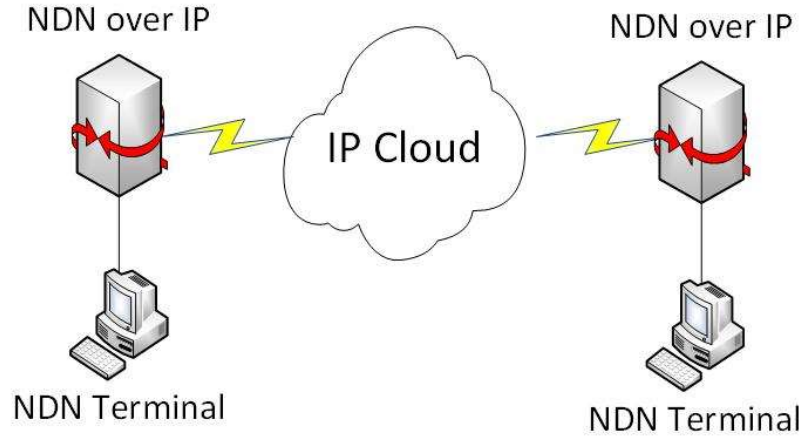
In [17], the authors showed a comparison of video streaming with different resolutions by using hICN and TCP connection. The caching feature in hICN had proven working correctly and improved the network quality in this scenario.

## 2.2 Encapsulation approach

Encapsulation is the most well-known and instant approach to migrate from a TCP/IP to an NDN protocol. This kind of approach can be distinguished furthermore into two types of encapsulation.

### 2.2.1 Upper-layer encapsulation

The upper-layer encapsulation uses the TCP/IP as the base layer and the NDN as the upper layer. This type of encapsulation is the instant deployment approach to be implemented because the network is still dominated by TCP/IP protocol up to now.



**Figure 2-3:** ICN/NDN over IP

Upper layer encapsulation uses IP payload to forward the whole NDN packet. Therefore, the encapsulation approach does not require router upgrading. However, the encapsulation approach still requires a pair of the gateway, as shown in Figure 2-3. Even though encapsulation is the most practical way to migrate from IP to NDN, there is some trade-off that possibly occurred. Packet header overhead is one of the disadvantage from this approach.

### 2.2.2 Under-layer encapsulation

The under-layer encapsulation implements NDN as a base layer and IP as an upper layer. This approach obligates all the routers to be upgraded into ICN/NDN routers with an extra pair of gateways for encapsulation between IP terminal, as shown in Figure 2-4. The authors think that the under-layer encapsulation is an unrealistic scenario for IP to ICN/NDN migration since the goal of migration is to enable ICN/NDN protocol over the IP protocol, not the opposite manner.

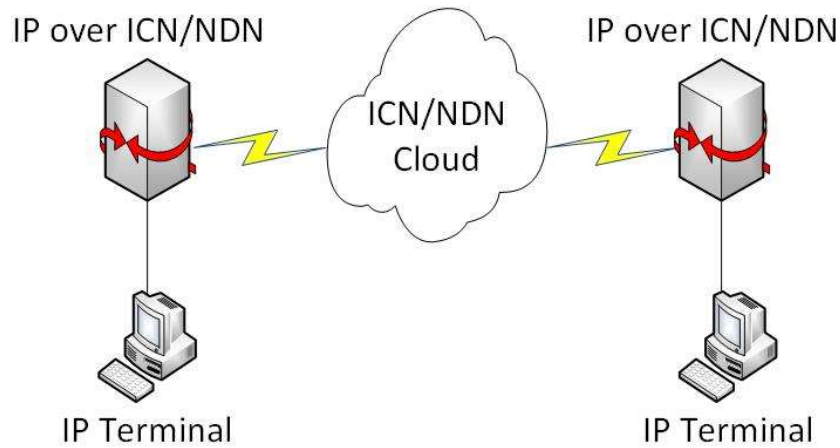


Figure 2-4: IP over ICN/NDN

### 2.3 Translation approach

Translation is the mechanism of converting from TCP/IP to ICN/NDN and vice versa. It considers as an economical migration approach to be implemented without losing the merit of ICN/NDN protocol in [6, 34, 36, 23, 28]. This type of migration demands a gateway used as an interpreter between the two network protocol, as shown in Figure 2-5. Referring to the Open System Interconnection (OSI) layer, the translation approach can be divided into three levels of translation.

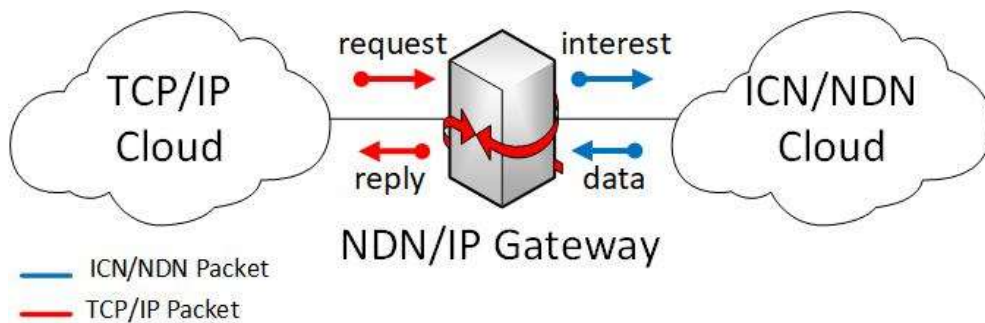


Figure 2-5: Translation gateway

### **2.3.1 Network layer translation**

The network layer translation focuses on translating the difference semantic primitives between IP and ICN/NDN protocol. The IP protocol consists of two semantic primitives, namely sending (push) and receiving (pull) while the ICN/NDN has only one semantic primitive called as a pull-driven semantic. The semantic difference causes a complex translation mechanism in the gateway. Furthermore, network layer translation should also considers the differences of a packet type like interest and data packet in ICN/NDN protocol. And how the packet is named independently whether using a name resolution service in the gateway or using other method.

### **2.3.2 Transport layer translation**

The transport layer translation deals with the handshaking and sequencing process that occurred in TCP and UDP packet of IP protocol. The other translation is how to translate the TCP handshake mechanism and congestion control into ICN/NDN packet. The transport layer translation treats a big-size data as one unit of packets. Thus, a hierarchical naming scheme is used for naming packet sequence.

### **2.3.3 Application layer translation**

The application layer translation works with many semantic existed in IP based application. The semantic translation is varied and exclusive for each application. For instance, the HTTP commands such as GET, HEAD, PUT, DELETE, POST, TRACE, and OPTIONS in TCP/IP must be translated into the ICN/NDN semantic. As a result, the name resolution sets a hierarchical naming such as application ID, command ID, and content ID for naming a prefix name. The application layer translation is the most complicated translation due to application command diversity.

In general, IP to ICN/NDN translation approach requires a naming resolution mechanism as part of its translation process. This mechanism is responsible for nam-

ing each packet arriving in the gateway. Every layer translation has its characteristic for naming the packet. For example, application layer translation has a longer prefix name compared with other layers. The network layer translation has a shorter prefix name due to packet independence and flexibility.

## 2.4 Migration cost

The expense of new hardware purchases should be minimal to avoid up-front capital investment because of migration process. On the other hand, the total cost of migration depends on the cost of the individual components, as reported in [9]. The authors of [9] described that the total estimation of migration cost,  $C_t$ , for  $L$ , the number of the routers, was formulated by

$$C_t = \sum_{i=1}^L \{\xi_a(C_s + \xi_b C_h) + (-\xi_a)C_r + C_v + C_d + C_m\}, \quad (2.1)$$

where the symbols of cost metrics are described in Table 2.1.

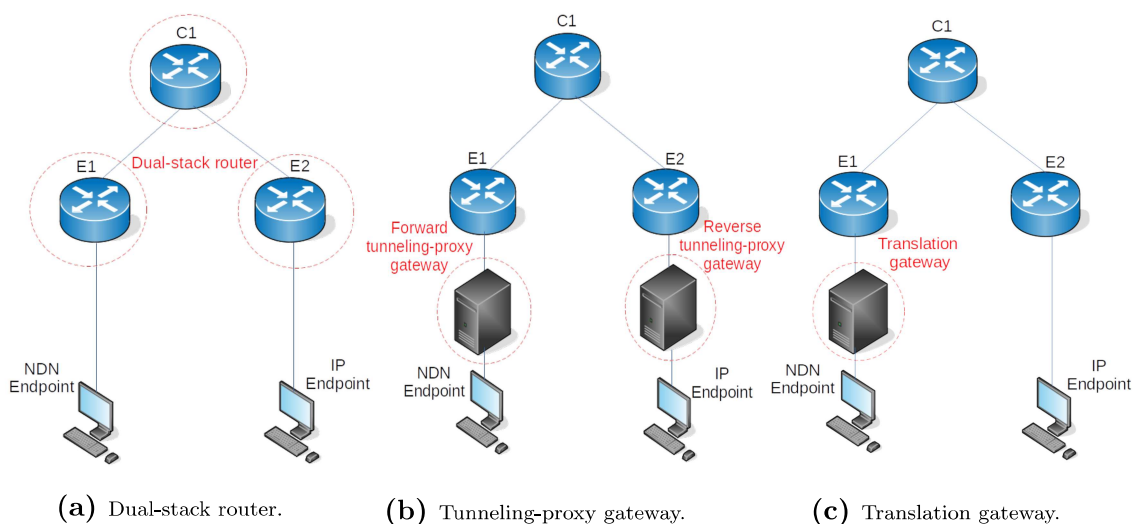
**Table 2.1:** Symbols related to cost metrics.

Symbol	Description
$C_s$	Software upgrade cost
$C_h$	Hardware upgrade cost
$C_r$	Hardware replacement cost
$C_v$	Vendor support cost
$C_d$	Development cost
$C_m$	Miscellaneous cost
$\xi_a$	Decision coefficient of software upgrade
$\xi_b$	Decision coefficient of hardware upgrade
$-\xi_a$	Negation of software upgrade decision coefficient

The symbol  $\xi_a$  and  $\xi_b$  have Boolean values of either 1 (True) or 0 (False). If the value  $\xi_b$  equals 1, then a router is upgraded. While  $\xi_b$  equals 0, then it means otherwise. However, the necessity of a router upgrade is influenced by hardware



peripherals such as memory, network interface, and processor speed. Under certain conditions, it is impossible to upgrade the hardware without a software upgrade. Thus, if  $\xi_b$  equals 1, then  $\xi_a$  is likewise. On the other hand, the hardware upgrade is non-compulsory following the software upgrade. Hence,  $\xi_b$  can be either 0 or 1, although  $\xi_a$  is equal to 1. Furthermore, the router replacement means the investment of new hardware with the latest pre-installed software. Therefore, the equation 2.1 has symbol  $-\xi_a$  multiplied by  $C_r$  which is the negation from  $\xi_a$  multiplied by  $C_s$  and  $C_h$ . This is because the router replacement will automatically nullify the hardware or software upgrade and vice versa.

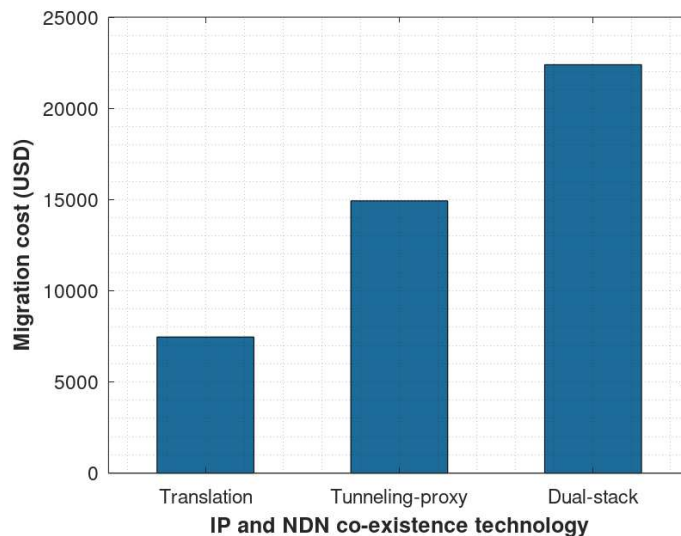


**Figure 2-6:** IP and NDN co-existence technology.

**Table 2.2:** Average migration cost of individual components.

Description	Cost
Software upgrade cost	\$ 275
Hardware upgrade cost	\$ 700
Hardware replacement cost	\$ 7000
Vendor support cost	\$ 140
Development cost	\$ 250
Miscellaneous cost	\$ 75

To compare migration costs on different IP and NDN co-existence technology, we consider a simple network topology consisting of three routers, namely  $C1$ ,  $E1$ , and  $E2$ , as shown in Figure 2.6. Afterward, we evaluate the total migration cost using the average migration cost of individual components in [9], as described in Table 2.2.



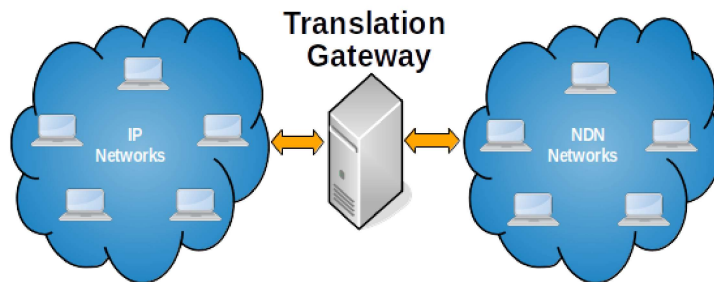
**Figure 2.7:** Migration cost in IP and NDN co-existence technology.

The result shows that the migration cost of the dual-stack router is the most expensive, followed by the tunneling-proxy and translation gateway, as seen in Figure 2.7. This is because the dual-stack router deployment demands total router replacements. On the other hand, the tunneling-proxy gateway necessitates two additional hardware installations on each different network protocol, while the translation gateway requires only one additional hardware on the border of the NDN network.

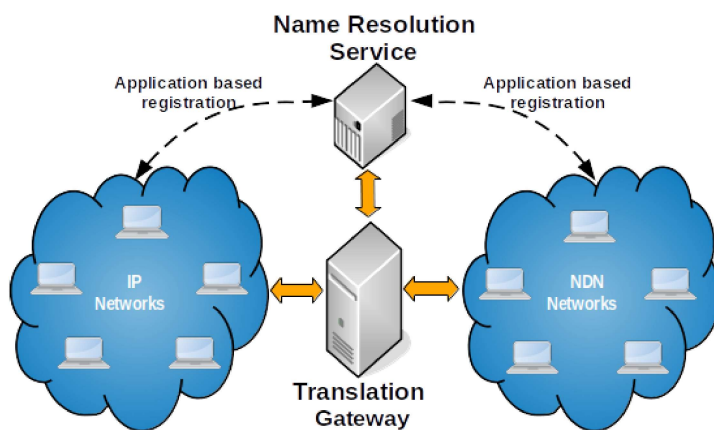
## Chapter 3

# The Dual-Channel Translation Gateway

This chapter describes the principles, crucial components, packet flows of the proposed dual-channel IP-to-NDN translation gateway. Furthermore, we propose two possible deployment schemes for prefix-name binding in the translation: static prefix-name binding and dynamic prefix-name binding.

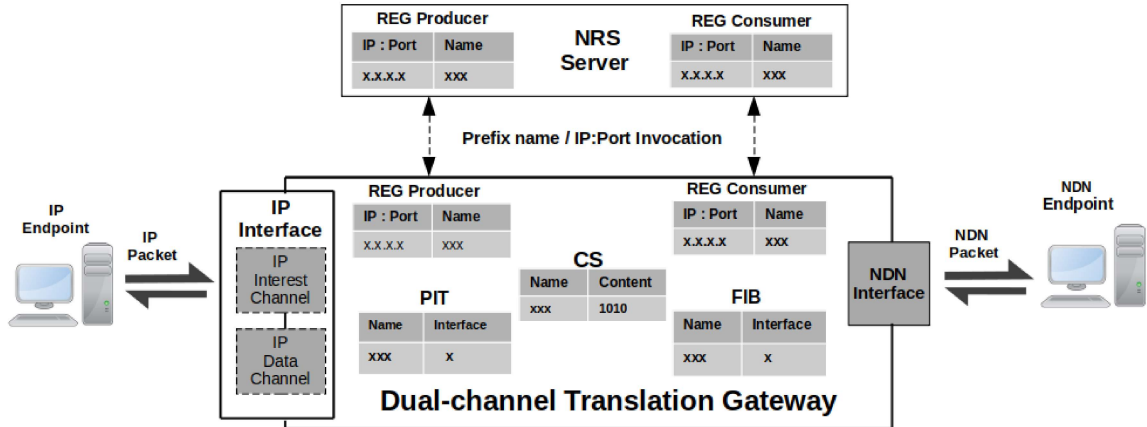


(a) Static prefix-name binding.



(b) Dynamic prefix-name binding.

**Figure 3-1:** Dual-channel IP-to-NDN translation gateway and its prefix-name binding.



**Figure 3-2:** Components of dual-channel IP-to-NDN translation gateway.

### 3.1 Principles

The fundamental concept of the dual-channel IP-to-NDN translation gateway is based on two distinct channels and a flexible-length packet naming strategy. The gateway assigns each incoming or outgoing IP packet to an allocated unique IP address. One unique IP address is associated with a corresponding interest packet type, while the other is associated with a corresponding data packet type. When an IP packet is received via the unique IP address attached to the interest type, the gateway immediately recognizes it as an interest packet from the IP consumer. If, however, it is received over the IP address attached to the data type, the gateway will immediately recognize it as a data packet from the IP producer.

Regarding the variable length of the packet naming strategy, a table called a register table (REG) is required in the dual-channel gateway so that each IP packet in the gateway is entitled to a prefix name. Since two distinct situations of producer and consumer exist between IP and NDN, the gateway must have two asymmetric REG tables, namely REG producer and REG consumer. When an IP endpoint acts as a consumer, the REG consumer table is utilized to obtain a specific prefix name.

However, when an IP endpoint acts as a producer, the REG producer table is used instead. Furthermore, we consider the independence of the REG producer and REG consumer tables because an IP endpoint may attach to a different prefix name when it is regarded as a producer or consumer in the IP networks.

The gateway registers all the prefix names in the static prefix-name binding in advance into the REG producer table in case of IP terminal as producer and NDN terminal as consumer and into REG consumer table in case of IP terminal as consumer and NDN terminal as producer. Therefore, when the gateway cannot resolve the prefix name from a given IP address, it drops immediately without allowing the client to modify the REG database on the fly. We assume that this is suitable for small-scale network deployment as seen in Figure 3-1(a). On the other hand, in dynamic prefix-name binding, the gateway utilizes an outer name server that feeds the gateway REG table with a set of associations of prefix names and IP addresses when requested. We use the Name Resolution Service (NRS), proposed in [21] and [24], to provide the packet-to-name binding service for the dual-channel translation gateway. Due to limited storage capacity, the gateway partially stores a set of associations. However, the NRS must preserve all prefix names in the set of associations in the network. It can alter the set of associations when a client in IP or NDN networks registers a new prefix name to bind with a particular IP address through an application-based service. This is called dynamic prefix-name binding, as seen in Figure 3-1(b).

## 3.2 Components of dual-channel gateway

The important components of the dual-channel IP-to-NDN translation gateway illustrated in Figure 3-2 can be explained as follows.

- **IP interest channel**

The gateway utilizes a static IP address to recognize IP interest-like packets

between the IP endpoint and the gateway. The static IP address used as the interest channel is known in advance by all IP endpoints in the network.

- **IP data channel**

The gateway has a dedicated static IP address for sending and receiving IP data packets. All IP endpoints in the network recognize the static IP address chosen as the data channel in advance as well.

- **NDN interface**

The NDN interface is constructed using ethernet layer interfaces. The NDN packet format follows the NDN packet v.3 specifications.

- **REG**

REG is used to provide information about the prefix name associated with the combination of IP address and port number. It consists of two different independent tables, namely a producer table and consumer table.

- **PIT**

PIT is a native NDN router component that is used as a pool for collecting information regarding the incoming interfaces of interest and its corresponding prefix name. If the requested prefix name content is replied to by the producer, the PIT table will eliminate the prefix name and its associated interfaces.

- **FIB**

FIB is used for forwarding an interest packet to its destination by retrieving interfaces that are associated with its particular prefix name. Afterward, the router forwards the interest packet through those interfaces.

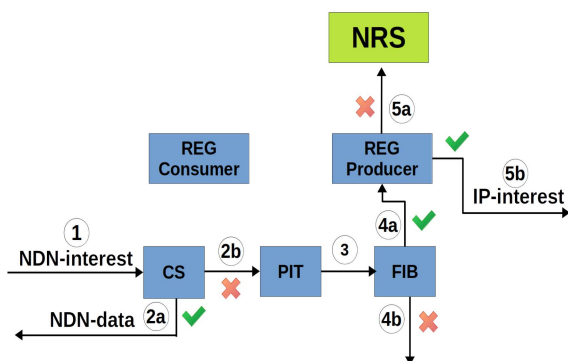
- **CS**

CS has a function to store temporally the received data packets that are passed

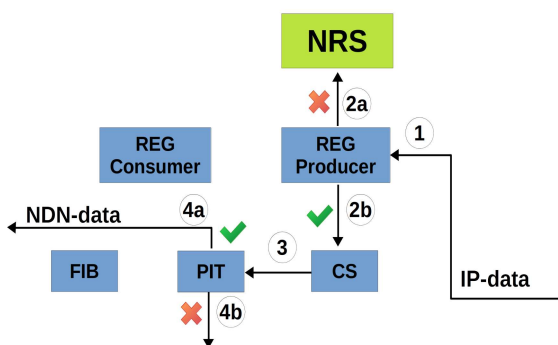
through the NDN router from a producer. The cached data content is forwarded later on to the consumer that requested the same prefix name.

### 3.3 Packet flows and translation procedure

This section describes the IP-to-NDN packet translation flow. The packet flow of the gateway differs in two scenarios. The packet flow in the case of the IP endpoint becoming a producer and NDN endpoint becoming a consumer, denoted as *scenario 1*, is shown in Figure 3-3. Alternatively, Figure 3-4 shows the packet translation flow in the case of a scenario where an NDN endpoint becomes a producer, and an IP endpoint becomes a consumer, denoted as *scenario 2*.



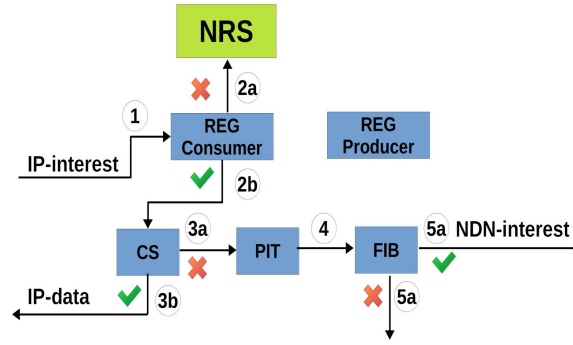
(a) NDN interest to IP interest packet flow.



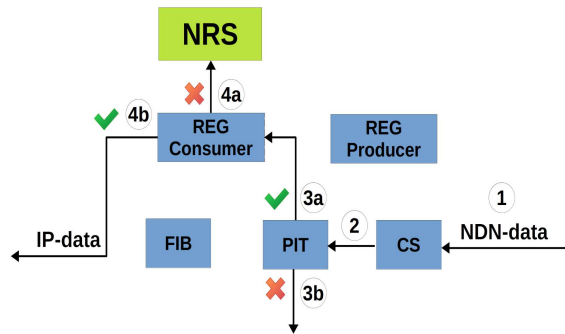
(b) IP data to NDN data packet flow.

**Figure 3-3:** Packet flow translation between IP and NDN protocol in the case of scenario 1.

In the case of scenario 1, when a gateway receives an interest packet (1), it checks its CS for that packet availability, whether the content exists or not. The gateway responds with an NDN data packet if it is available (2a). If the requested content does not exist in the CS, the interest packet is passed to PIT, which registers the incoming interface (2b), and the gateway checks the forwarding interface at FIB (3). In the dual-channel gateway, FIB must also communicate with REG producer table before forward the interest packet. This step is necessary to ensure that the producer is located in the IP networks. FIB drops the packet if there is no information about the prefix name being searched for (4b). When the outgoing interface is discovered and located in the IP network, it is passed to the REG producer table to determine



(a) IP interest to NDN interest packet flow.



(b) NDN data to IP data packet flow.

**Figure 3.4:** Packet flow translation between IP and NDN protocol in the case of scenario 2.



---

**Algorithm 1** Dual-channel packet translation on IP interface.

---

**Input:** IP packet

**Output:** IP packet or NDN packet

*LOOP Process*

```
1: if IP interest channel receives packet then
2:   Retrieve prefix name in REG Consumer
3:   while Prefix name is not in REG Consumer do
4:     Request update and wait response to NRS server
5:   end while
6:   if Content is available in CS then
7:     Retrieve content and send via IP data channel
8:   else
9:     Name-based routing (Algorithm 3) via NDN channel
10:  end if
11: end if
12: if IP data channel receives packet then
13:   Retrieve prefix name in REG Producer
14:   while Prefix name is not in REG Producer do
15:     Request update and wait response to NRS server
16:   end while
17:   Content caching (Algorithm 4) via NDN channel
18: end if
```

---

the corresponding IP address (4a). If it is not found in the IP network, the interest is forwarded to the interface that connect to NDN network. The gateway sends a resolving packet to NRS and waits for a response because the REG producer table is likewise restricted (5a). The IP interest packet is then transmitted across the IP channel (5b) as seen in Figure 3-3(a). Figure 3-3(b) shows the processing flow at the gateway when the data packet arrives at the gateway from an IP provider. When an IP data packet sent from an IP producer arrives at the gateway, the gateway checks the IP address and port number combination in the REG producer table (1). If it is not found, the gateway sends a resolving packet to NRS (2a). It then forwards the data packet through the interface (4a) before it is found in the PIT table (3) and stored in CS (2b).

The packet flow translation in the case of scenario 2 is quite similar to scenario 1, except that the interest packets are first checked in the REG consumer table to determine the associated prefix name (1). When an IP packet is received from NRS, it is converted to an NDN packet and then checked for availability in the CS (2b). If it is not found, the packet is sent to the NDN interface (5a) that was previously

---

**Algorithm 2** Dual-channel packet translation on NDN interface.

---

**Input:** NDN packet**Output:** NDN packet or IP packet*LOOP Process*

```
1: if NDN channel receives NDN interest packet then
2:   if Content is available in CS then
3:     Retrieve content and send via via NDN channel
4:   else
5:     Retrieve IP address and port in REG Producer
6:     while IP address and port is not in REG Producer do
7:       Request update and wait response to NRS server
8:     end while
9:     Name-based routing (Algorithm 3) via IP interest channel
10:  end if
11: end if
12: if NDN channel receives NDN data packet then
13:   Retrieve IP address and port in REG Consumer
14:   while IP address and port is not in REG Consumer do
15:     Request update and wait response to NRS server
16:   end while
17:   Content caching (Algorithm 4) via IP data channel
18: end if
```

---

found in the FIB (4) and registered in the PIT incoming interface (3a) as seen in Figure 3-4(a). When the data packet has arrived at the gateway, as shown in Figure 3-4(b), the gateway stores the content in CS (1) and verifies the outgoing interface in PIT (2) when the NDN producer delivers the NDN data packet. The prefix name is used to retrieve the related IP address in the REG consumer table because the outgoing interface is an IP interface. In the event that it is not found, the gateway sends a REG resolving packet to NRS server (4a). The data content is transformed into an IP packet after receiving a response from the NRS server. The data packet is subsequently forwarded across the IP data channel (4b).

---

**Algorithm 3** Name-based routing.

---

**Input:** Interest packet**Output:** Interest packet

```
1: Register incoming interface in PIT
2: Retrieve outgoing interface in FIB
3: if Prefix name is not in FIB then
4:   Drop interest packet
5: else
6:   Forward interest packet
7: end if
```

---

The dual-channel translation gateway operates on two network interfaces that continuously listen for incoming packets. Algorithm 1 illustrates the pseudo-code for

---

**Algorithm 4** Content caching.

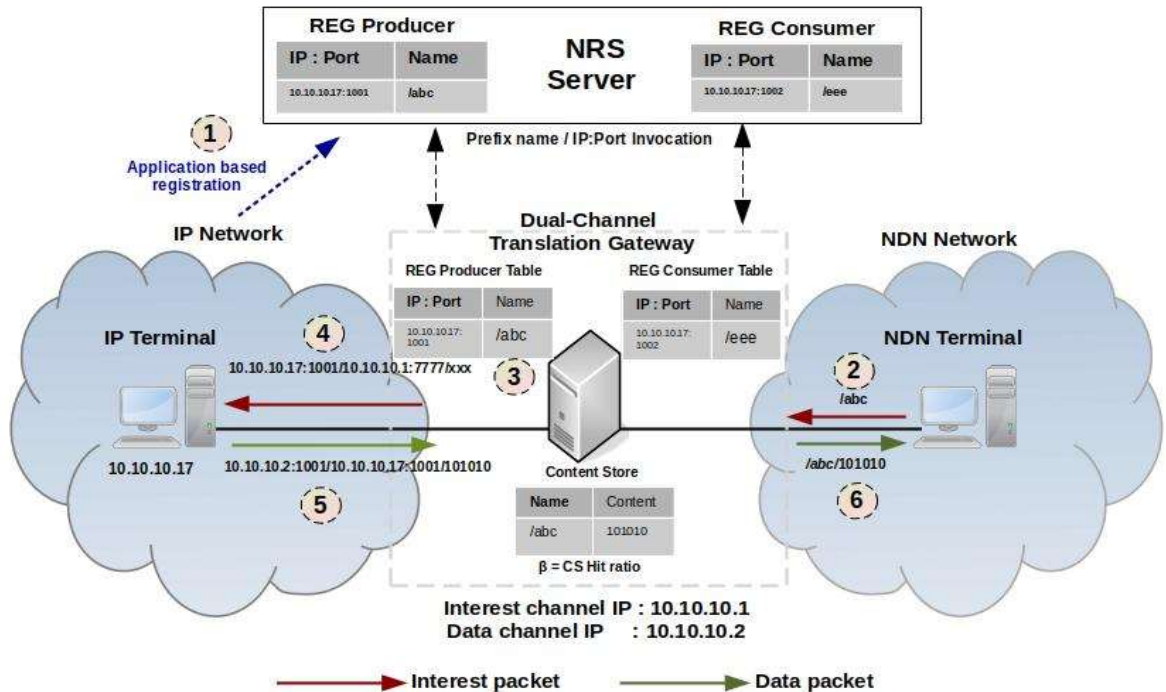
---

**Input:** Data packet  
**Output:** Data packet  
1: Store data content to CS  
2: Retrieve outgoing interface in PIT  
3: **if** Prefix name is not in PIT **then**  
4:     Drop data packet  
5: **else**  
6:     Delete PIT entry  
7:     Forward data packet  
8: **end if**

---

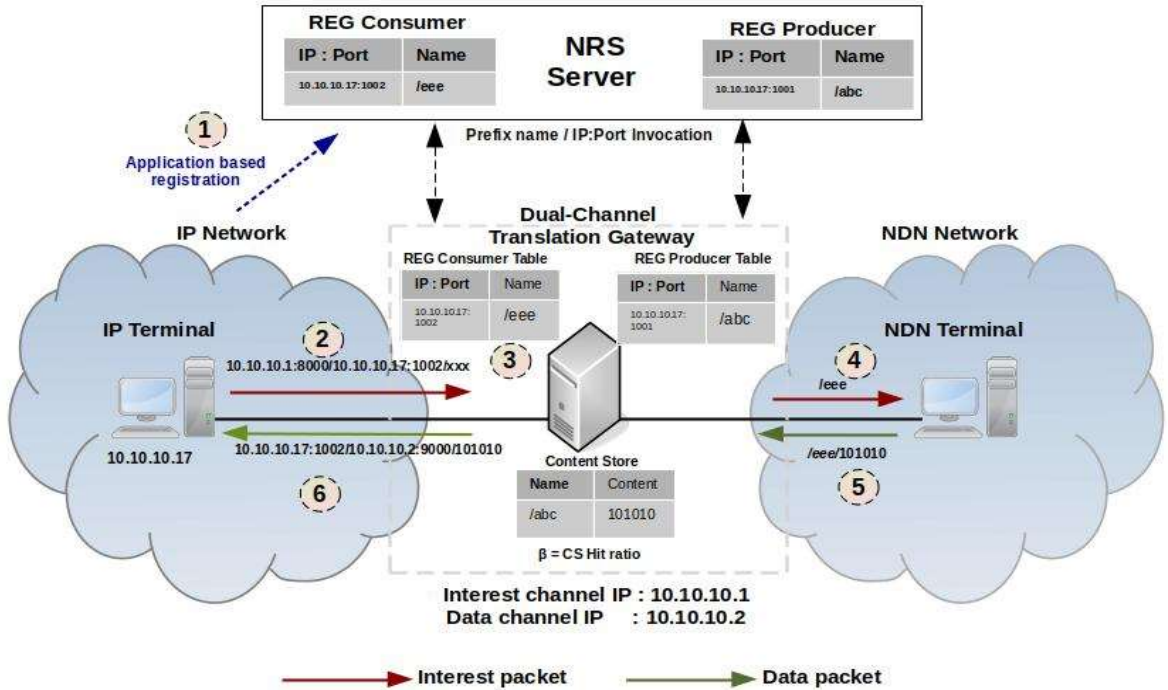
the dual-channel gateway translation procedure on the IP interface. Alternatively, Algorithm 2 describes the pseudo-code for the translation procedure on the NDN interface. We define two native procedures of ICN/NDN routers, namely named-based routing and content caching, to simplify the instructions of Algorithm 1 and 2. Name-based routing forwards the interest packet to the producer as shown in Algorithm 3. On the other hand, content caching stores the data packet to CS and forwards it to the consumer as described in Algorithm 4. In the initialization setup, two unique IP addresses are configured to the IP interface: IP interest and data channel, while the NDN interface is attached to a specific MAC address. When an IP interface receives an incoming packet, the gateway examines if it is of interest or a particular data type. The NDN interface is likewise subjected to the exact mechanism. Assume that the NDN interface receives packets. The gateway verifies the specific type of packet and applies infinite looping to keep the process still operating on both network interfaces.

To give more explanation, we consider the scenario where the NDN node becomes a consumer, and the IP node acts as a producer using an example shown in Figure 3-5. We assume the NRS server possesses all mapping entities between prefix names to a combination of IP address and port numbers of the IP terminals active as producer node. The IP terminal registers a prefix name mapping to a combination of IP addresses and the port number via application-based service to the NRS server in advance (1). First, the NDN node emits an interest packet with a particular prefix



**Figure 3-5:** Translation process in the scenario 1

name, /abc, to the gateway (2). Next, the gateway tries to translate the NDN interest packet to the IP interest packet by retrieving a combination of an IP address and a port number for the corresponding prefix name in the REG producer table (3). The gateway constructs an IP packet with a destination address header of 10.10.10.17:1001 and a source address header of 10.10.10.1:7777. Afterwards, the gateway sends it through the interest channel (4). IP node replies directly by sending the content with the IP packet with source address 10.10.10.17:1001 through the data channel with destination address header 10.10.10.2:1001 (5). Once again, the gateway receives the packet, finds the prefix name from the source address using the REG producer table. Finally, the NDN data packet with the prefix name /abc is sent to the NDN client and construct an NDN data packet following the standard NDN data packet forwarding process (6).



**Figure 3-6:** Translation process in the scenario 2

On the other hand, we consider the scenario where the IP node becomes a consumer and the NDN acts as a producer using an example shown in Figure 3-6. We assume the NRS server's REG consumer table shows a mapping between prefix name and a combination of IP address and port number of the request IP terminal. In particular, prefix name /eee is mapped to a source IP address 10.10.10.17:1002. The map table mechanism is similar to DNS in the IP network that registers in advance based on the application-registration mechanism (1). The consumer initiates by sending an interest-like IP packet. First, the IP node constructs an IP header packet with 10.10.10.1:8000 in the destination part and 10.10.10.17:1002 in the source part with any ignored data payload (2). The gateway knows that the packet is an interest packet when receiving this packet since it uses IP 10.10.10.1 as the destination address. The gateway converts this packet into an NDN packet with prefix name mapped in REG

consumer table from the source IP address and port number (3). The NDN producer receives the interest packet with the prefix name /eee (4) and replies directly by sending the data packet to the gateway (5). The gateway translates the packet by constructing the IP packet with 10.10.10.17:1002 as the destination address and sends it through a data channel (6).

### 3.4 User privacy and security

The NDN technology uses a prefix name in its interest or data packets. A prefix name is written out in plain-text so that a consumer can easily identify the name of the content. The content in the data packet can be signed and encrypted by exchanging keys between producer and consumer. On the other hand, IP protocol has source and destination addresses in its packets that propagate through routers until they reach their final destination. Unless the data is encrypted at the application layer, IP data packets are usually sent in plain text form. It is challenging to maintain security concerns across those protocols due to the distinct behavior of exposed packet components.

The gateway relies on data packet load to transport other protocols from node to node in the encapsulation approach. The NDN packet is placed inside the IP data load in the case of NDN over IP. The unencrypted data payload may disclose both the NDN prefix name and the content within the IP network. When the packet propagates from router to router, the interceptor can easily determine the prefix name and the content. The attacker may also modify the content resulting in a data integrity breach called a Content Poisoning Attack (CPA). The CPA has been reported as the main cause of the Denial of Service (DoS) attack in the NDN network [32].

The dual-channel gateway can lower the risk of a CPA attack by separating the prefix name and the data content. The prefix name is located in the REG table in the

gateway linked to one of the IP headers, either the destination or source address. The propagated IP packet does not expose the prefix name inside the IP data payload. Therefore, the CPA can be reduced in the dual-channel gateway. At the same time, it also guarantees user privacy in the translation process at the gateway. This is because the gateway does not require reading the IP data payload content bit by bit. The translation mechanism in the dual-channel only necessitates the IP address header reading.

### **3.5 Prefix name preregistration**

As mentioned in the previous section, IP and prefix name preregistration is necessary and must be implemented before sending interest to the dual-channel gateway. In the case of static prefix name binding, the administrator manually registers either the REG producer or REG consumer table. Therefore, the IP endpoint and NDN endpoint cannot modify the IP address and port number binding to the prefix name in an instance. On the other hand, in the case of dynamic prefix name binding, the IP or NDN endpoint can modify the IP address and port number to prefix name binding on the fly since the binding registration can be done anytime before initiating the communication to the translation gateway. The registration process utilizes a global name mapping server, Name Resolution Service. NDN endpoint does not need to register at NRS. Instead, it need to register in routing protocol

#### **3.5.1 Name resolution service**

Several authors propose a global name resolution service, NRS, in the ICN/NDN network as reported by Liu in [21] and Hong in [24]. The NRS is a database of global content's names in the ICN network. The purpose of an NRS in ICN is to translate an object name into other information, such as a locator, another name, etc., to forward the object request. Moreover, Luo et al. extend the NRS server functionality for IP

and ICN/NDN co-existence in [45]. The NRS server must implement the dual-stack protocol for deploying IP and ICN/NDN co-existence. Currently, many researchers still actively discuss the development of NRS to standardize the implementation of this function. Therefore, this manuscript briefly describes the NRS as a fundamental requirement for the IP-to-NDN translation process. Additionally, the function should be leveraged with IP address and port number binding registration and query service. This function is mandatory in dynamic prefix name binding for the dual-channel IP-to-NDN translation gateway.

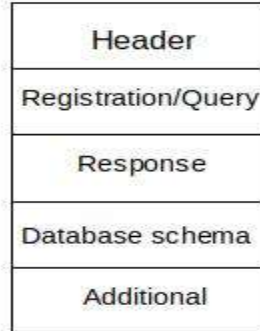
### 3.5.2 Registration packet format

Registration packet format must be defined to provide registration and query for prefix name binding. This service uses application-based registration to the NRS server. The system provides general name query interfaces. For the given NDN prefix type, it returns a resource record set. As for the given associated IP address and port number, it returns a prefix name. The system also provides interfaces for dynamic updates. In general, the application must provide the following:

- Mechanism for registering the prefix name and its associated IP address and port number.
- Mechanism for querying the prefix name and its associated IP address and port number.
- Mechanism for responding to query and registration.
- Mechanism for maintaining security in its registration and querying process
- Schema for the database selection either REG producer or REG consumer.

All NRS packets have a structure that is a header, registration/question, answer, and additional, as seen in Figure 3-7. The header describes the type of packet and





**Figure 3.7:** NRS packet format

which fields are contained in the packet, such as it is registration or query, number of registration/queries, time-to-live, certificates, etc. It is also possible that a response to a single question may contain multiple answers, such as if a name has multiple IP addresses and port numbers. Alternatively, It is also possible that a client registers multiple addresses and prefix names at a single invocation to the NRS server.

### 3.5.3 Prefix name registration in scenario 1

In the case of scenario 1, the IP terminal becomes a producer, and the NDN terminal becomes a consumer. Thus, the IP terminal should initially register the IP address and port number to the REG producer table. As static registration, the registration can be executed by an administrator that can directly access and modify the REG table at the dual-channel translation gateway. However, in the dynamic prefix name binding, the registration should be done by the IP terminal to the NRS server so that it can feed the information to the dual channel whenever the necessary information has not been found in the REG producer table.

The IP terminal sends a registration packet to the NRS server consisting of the IP address, port number, prefix name, and REG producer table or its database schema. Multiple registrations can be done for a set of combination IP addresses and port numbers with NDN-style prefix names to the NRS server by the IP producer so that

it can optimize the registration time. Afterwards, the IP producer is ready to accept the IP-interest packet from the dual-channel translation gateway since a binding mechanism has been completed. From the NDN endpoint, the NDN consumer needs to send an interest packet with a specific prefix name that associates with a particular IP address and port number in the IP producer without any registration process.

#### **3.5.4 Prefix name registration in scenario 2**

In the case of scenario 2, the IP terminal becomes a consumer, and the NDN terminal becomes a producer. The IP endpoint must register its IP address and port number and the prefix name of its accessing content to the REG consumer table before sending an interest packet to the translation gateway. Similar to scenario 1, an administrator can only execute a static registration. On the other hand, in the dynamic prefix name binding, the registration is executed by the IP terminal to the NRS server so that it can provide the necessary information to the dual channel whenever the requested prefix name is not found in the REG consumer table.

The IP terminal sends a registration packet that provides a set of IP addresses and port numbers with NDN-style prefix names, database schema, lifetime, and security signature. The registration of the content name in the NDN network can be done automatically by Name Data Link State Routing Protocol (NLSR) or other application-based mechanisms since it is still evolving up to now. Multiple registrations of combination IP address: port and prefix name to NRS server is also possible. For example, an IP client sends a registration to the NRS server consisting of a set of registration information. A set of registration information can be constructed by binding the IP client's address and several port numbers to different content names. Therefore, an IP client requires to register once to request several content names to the NRS server, which will save the registration time. Afterwards, the IP consumer is ready to send the IP-interest packets to the dual-channel translation gateway since

a binding mechanism has been completed. From the NDN network side, the NDN producer just needs to respond to interest packets with data packet with requested content.

## Chapter 4

# Throughput Estimation Model

This chapter describes the throughput estimation model of the proposed dual-channel IP-to-NDN translation gateway. The model identifies that throughput is the product of cache hit ratio and packet processing time in general. The behavior of name resolution table and its impact to the throughput is also analyzed

### 4.1 Throughput Analysis

Throughput is defined as the average number of data bits transmitted in a second. Thus, we define throughput as the average packet size divided by  $\tau$ , where  $\tau$  is the average time of a request that has elapsed in the gateway. As shown in Figure 3-3 and 3-4, the value of  $\tau$  depends on the schemes, scenarios, and cases of packet flow at the gateway. In the case of dynamic prefix-name binding scheme, scenario 1 has five different cases of packet flow, whereas scenario 2 has six different cases. This is because the combination of CS and REG affects the number of packet flow cases. However, in the case of static prefix-name binding, the number of packet flow cases is reduced to 2 for both scenarios 1 and 2. This is because only the CS component affects the packet flow translation.

The dynamic prefix-name binding scheme allows clients to dynamically register prefix names through a name server. Therefore, the NRS contains all prefix-name bindings in the network, while the translation gateway partially stores the prefix-name bindings in a cache due to its limited memory capacity. A cache miss in the

translation gateway will cause a request to be made to the NRS for a target prefix name. As a result, in addition to  $\beta$ , CS hit ratio, there is a supplemental parameter for a prefix-name binding cache called  $\gamma$ . We introduced  $\gamma_{p,1}$  as the hit ratio of the REG producer table (REGp) and  $\gamma_{c,1}$  as the hit ratio of the REG consumer table (REGc) in the case of requesting IP address by invoking a prefix name. Alternatively,  $\gamma_{p,2}$  is the hit ratio of the Reverse REG producer table (RREGp), and  $\gamma_{c,2}$  is the hit ratio of the Reverse REG consumer table (RREGc) in the case of requesting a prefix name by invoking an IP address. All caches in the dual-channel IP-to-NDN translation gateway, such as CS and REG table, used an LRU cache replacement algorithm in this manuscript unless it is mentioned otherwise. Table 4.1 shows a description of delay components in the gateway used in the throughput model.

**Table 4.1:** Definition of delay components.

Symbol	Description
$t_c$	Positive look-up time in CS
$t_{cn}$	Negative look-up time in CS
$t_{reg1}$	Positive look-up time in REGp or REGc
$t_{reg1'}$	Negative look-up time in REGp or REGc
$t_{reg2}$	Positive look-up time in RREGp or RREGc
$t_{reg2'}$	Negative look-up time in RREGp or RREGc
$t_{ip}$	NDN interest packet parsing time
$t_{is}$	NDN interest packet serialization time
$t_{dp}$	NDN data packet parsing time
$t_{ds}$	NDN data packet serialization time
$t_{ipc}$	IP packet construction time
$t_{ipd}$	IP packet deconstruction time
$t_{pr1}$	Response time from IP producer
$t_{pr2}$	Response time from NDN producer
$t_{soh}$	Socket overhead time
$t_{nrs}$	NRS resolving round trip time

#### 4.1.1 Static prefix-name binding

When static prefix-name binding is implemented in the network, the hit ratio of CS is the dominant parameter that affects the overall throughput. As for scenario 1,

the value of  $\tau$  can be described by

$$\tau = \begin{cases} ts_{1a} & \text{if CS hit} \\ ts_{1b} & \text{if CS miss} \end{cases}$$

where

- $ts_{1a} = t_{i_p} + t_c + t_{soh}$ ,
- $ts_{1b} = t_{cn} + t_{i_p} + t_{ip_c} + t_{ip_d} + t_{d_s} + t_{pr1} + 2t_{soh}$ .

However, in the case of scenario 2, the value of  $\tau$  can be defined by

$$\tau = \begin{cases} ts_{2a} & \text{if CS hit} \\ ts_{2b} & \text{if CS miss} \end{cases}$$

where

- $ts_{2a} = t_c + t_{i_p} + t_{ip_c} + t_{ip_d} + t_{d_p} + t_{soh}$ ,
- $ts_{2b} = t_{cn} + t_{i_s} + t_{ip_c} + t_{ip_d} + t_{d_p} + t_{pr2} + 2t_{soh}$ .

Thus, the average processing time,  $\tau$ , in the case of scenario 1 can be expressed by

$$\tau = \beta ts_{1a} + (1 - \beta)ts_{1b}, \quad (4.1)$$

and for scenario 2,  $\tau$  can be given by

$$\tau = \beta ts_{2a} + (1 - \beta)ts_{2b}, \quad (4.2)$$

where  $\beta$  is the CS hit ratio.

The value of  $\beta$  can be approximated by calculating the hit ratio of each prefix name,  $\beta(i)$ . A particular prefix names,  $i$ , follows random variable so that  $i \in \{1, \dots, M\}$

where  $M$  is the total number of unique prefix names. The  $\beta$  can be expressed by

$$\beta = \sum_{i=1}^M q(i)\beta(i). \quad (4.3)$$

Furthermore, the ratio of request of each content,  $q(i)$ , follows the probability of density function from Zipf distribution that can be formulated by

$$q(i) = \frac{1}{i^\alpha \sum_{l=1}^M (\frac{1}{l})^\alpha} \quad (4.4)$$

The approximation of the hit ratio can be calculated by using Che's approximation, which was originally proposed by Che et al. [19]. Then, the hit ratio for a particular prefix name  $\beta(i)$  can be estimated by the following

$$\beta(i) = 1 - e^{-q(i)t_c}. \quad (4.5)$$

$t_c$  is the unique root of the equation

$$C = \sum_{i=1}^M (1 - e^{-q(i)t_c}), \quad (4.6)$$

where  $C$  is the storage capacity of CS since we use LRU as a replacement algorithm in our translation gateway.

#### 4.1.2 Dynamic prefix-name binding

When dynamic prefix-name binding is implemented in the network, the overall throughput are collaboration from the hit ratio of CS, REGp, RREGp in the scenario 1 and CS, REGc, RREGc in the scenario 2. Therefore, in the case of scenario 1, the

value of  $\tau$  can be described by

$$\tau = \begin{cases} td_{1a} & \text{if CS hit} \\ td_{1b} & \text{if CS miss, REGp hit, and RREGp hit} \\ td_{1c} & \text{if CS miss, REGp hit, and RREGp miss} \\ td_{1d} & \text{if CS miss, REGp miss, and RREGp hit} \\ td_{1e} & \text{if CS miss, REGp miss, and RREGp miss} \end{cases}$$

where

- $td_{1a} = t_{ip} + t_c + t_{soh}$ ,
- $td_{1b} = t_{ip} + t_{cn} + t_{reg1} + t_{ipc} + t_{pr1} + t_{ipd} + t_{ds} + t_{reg2} + 2t_{soh}$ ,
- $td_{1c} = t_{ip} + t_{cn} + t_{reg1} + t_{ipc} + t_{pr1} + t_{ipd} + t_{ds} + t_{reg2'} + t_{nrs} + 3t_{soh}$ ,
- $td_{1d} = t_{ip} + t_{cn} + t_{reg1'} + t_{nrs} + t_{ipc} + t_{pr1} + t_{ipd} + t_{ds} + t_{reg2} + 3t_{soh}$ ,
- $td_{1e} = t_{ip} + t_{cn} + t_{reg1'} + t_{nrs} + t_{ipc} + t_{pr1} + t_{ipd} + t_{ds} + t_{reg2'} + t_{nrs} + 4t_{soh}$ .

The average time for processing a packet,  $\tau$ , can be obtained by

$$\begin{aligned} \tau = & \beta td_{1a} + (1 - \beta)\gamma_{p,1}\gamma_{p,2}td_{1b} \\ & + (1 - \beta)\gamma_{p,1}(1 - \gamma_{p,2})td_{1c} + (1 - \beta)(1 - \gamma_{p,1})\gamma_{p,2}td_{1d} \\ & + (1 - \beta)(1 - \gamma_{p,1})(1 - \gamma_{p,2})td_{1e}. \quad (4.7) \end{aligned}$$



However, in the case of scenario 2, the value of  $\tau$  can be defined by

$$\tau = \begin{cases} td_{2a} & \text{if CS hit, RREGc hit} \\ td_{2b} & \text{if CS hit, RREGc miss} \\ td_{2c} & \text{if CS miss, REGc hit, and RREGc hit} \\ td_{2d} & \text{if CS miss, REGc hit, and RREGc miss} \\ td_{2e} & \text{if CS miss, REGc miss, and RREGc hit} \\ td_{2f} & \text{if CS miss, REGc miss, and RREGc miss} \end{cases}$$

where

- $td_{2a} = t_{ip_d} + t_{reg2} + t_c + t_{d_p} + t_{ip_c} + t_{soh}$ ,
- $td_{2b} = t_{ip_d} + t_{reg2'} + t_{nrs} + t_c + t_{d_p} + t_{ip_c} + 2t_{soh}$ ,
- $td_{2c} = t_{ip_d} + t_{reg2} + t_{cn} + t_{i_s} + t_{d_p} + t_{pr2} + t_{reg1} + t_{ip_c} + 2t_{soh}$ ,
- $td_{2d} = t_{ip_d} + t_{reg2'} + t_{nrs} + t_{cn} + t_{i_s} + t_{d_p} + t_{pr2} + t_{reg1} + t_{ip_c} + 3t_{soh}$ ,
- $td_{2e} = t_{ip_d} + t_{reg2} + t_{nrs} + t_{cn} + t_{i_s} + t_{d_p} + t_{pr2} + t_{reg1'} + t_{ip_c} + 3t_{soh}$ ,
- $td_{2f} = t_{ip_d} + t_{reg2'} + t_{nrs} + t_{cn} + t_{i_s} + t_{d_p} + t_{nrs} + t_{pr2} + t_{reg1'} + t_{ip_c} + 4t_{soh}$ .

Therefore, the average packet processing time,  $\tau$ , for scenario 2 can be expressed by the following

$$\begin{aligned} \tau = & \beta\gamma_{c,2}td_{2a} + \beta(1 - \gamma_{c,2})td_{2b} + (1 - \beta)\gamma_{c,1}\gamma_{c,2}td_{2c} \\ & + (1 - \beta)\gamma_{c,1}(1 - \gamma_{c,2})td_{2d} + (1 - \beta)(1 - \gamma_{c,1})\gamma_{c,2}td_{2e} \\ & + (1 - \beta)(1 - \gamma_{c,1})(1 - \gamma_{c,2})td_{2f}. \end{aligned} \quad (4.8)$$

The value of the hit ratio in CS,  $\beta$ , can be calculated obeying the same equation (4.3) for static prefix-name binding with its associated parameters such as the storage capacity of CS,  $C$ , the ratio of requests for content  $i$  sent by the consumer,  $q(i)$ , and the unique root of an equation,  $t_c$ . However, the value of the hit ratio in the REG producer or consumer table may differ due to the packet flow behavior in the translation process. As for scenario 1, the values of  $\gamma_{p,1}$  and  $\gamma_{p,2}$  are highly correlated with the CS miss ratio,  $(1-\beta)$ . Therefore, an identical formula can be used to estimate

both  $\gamma_{p,1}$  and  $\gamma_{p,2}$ . First, we introduce the ratio of uncached content  $i$ ,  $r(i)$  as the product from the miss ratio of CS,  $(1-\beta)$ , which can be approximated by the following

$$r(i) = q(i)(1 - \beta(i)). \quad (4.9)$$

By using equation (4.6), the unique root,  $t_r$ , for the REGp table can be obtained by replacing the size of CS,  $C$ , with the size of REG,  $R$ . Finally, the hit ratio of a particular prefix name  $i$  in REGp and RREGp,  $\gamma_{p,1}(i)$  and  $\gamma_{p,2}(i)$ , must be adjusted by a normalization factor as described by

$$\gamma_{p,1}(i) = \gamma_{p,2}(i) = \frac{1}{1 - \beta} (1 - e^{-r(i)t_r}). \quad (4.10)$$

Finally, the cumulative hit ratio of  $\gamma_{p,1}(i)$  and  $\gamma_{p,2}(i)$  from  $M$  total unique prefix names can be defined by the following

$$\gamma_{p,1} = \gamma_{p,2} = \sum_{i=1}^M r(i)\gamma_{p,1}(i). \quad (4.11)$$

As for scenario 2, the value of  $\gamma_{c,1}$  is correlated with the CS miss ratio,  $(1-\beta)$ , so it follows equations (4.9). And the value of  $\gamma_{c,1}$  for each content  $i$  can be described as follow

$$\gamma_{c,1}(i) = \frac{1}{1 - \beta} (1 - e^{-r(i)t_r}). \quad (4.12)$$

Thus, the total  $\gamma_{c,1}$  is formulated by

$$\gamma_{c,1} = \sum_{i=1}^M r(i)\gamma_{c,1}(i). \quad (4.13)$$

However, the value of  $\gamma_{c,2}$  is not affected by the CS miss ratio. This is because the distribution of a prefix name requested in RREGc and CS is identical to  $q(i)$ . The unique root,  $t_r$ , for the REGc table can be obtained by

$$R = \sum_{i=1}^M (1 - e^{-q(i)t_r}). \quad (4.14)$$

Therefore, the hit ratio of  $\gamma_{c,2}$  for particular content  $i$  can be formulated by

$$\gamma_{c,2}(i) = 1 - e^{-r^{(i)}t_r}. \quad (4.15)$$

And the cumulative hit ratio of  $\gamma_{c,2}$  from  $M$  total unique prefix names can be defined by the following

$$\gamma_{c,2} = \sum_{i=1}^M q(i)\gamma_{c,2}(i). \quad (4.16)$$

## Chapter 5

# Throughput Evaluation

This chapter describes the numerical evaluation of our estimation model for the dual-channel IP-to-NDN translation gateway. The analysis model is compared to emulation test bed result. The emulation test bed was constructed following the topology as shown in Figure 3.1(a) in the case of static prefix-name binding and Figure 3.1(b) in the case of dynamic prefix-name binding. Virtual box and python 3.7 were used in the emulation testbed by utilizing the external library of NDN packet v.3 in [40].

The interest packets were generated asynchronously, about ten thousand packets per second, to measure the throughput. The distribution of requested content followed a Zipf distribution with skewed parameter  $\alpha$  due to some studies reporting the request distribution of various types of digital content. Websites and user-generated videos followed the Zipf distribution as reported in [27] and [29]. Breslau et al. said that the request count of webpages obeyed the Zipf distribution with a parameter  $\alpha$  between 0.64 and 0.83 in [27]. Moreover, Mahanti et al. showed that it was between 0.74 and 0.84 in [3]. The request count of YouTube videos obeyed the Zipf distribution with a parameter  $\alpha$  about 0.8 in [29]. Therefore, we assumed that  $\alpha$  was between 0.7 and 1.5 in our testbed to cover all possible request distribution regarding content popularity on the internet.

We captured the data log from our emulator testbed about fifty thousand packets as a data sample for time evaluation. A specific script to measure and write time of each process was injected into the dual-channel translation source code. We repeated the data collection process at random about ten times to ensure that the data was consistent and statistically compliant. Additionally, we took the data sample every second within five minutes from the data log to evaluate the throughput.

## 5.1 Static prefix-name binding

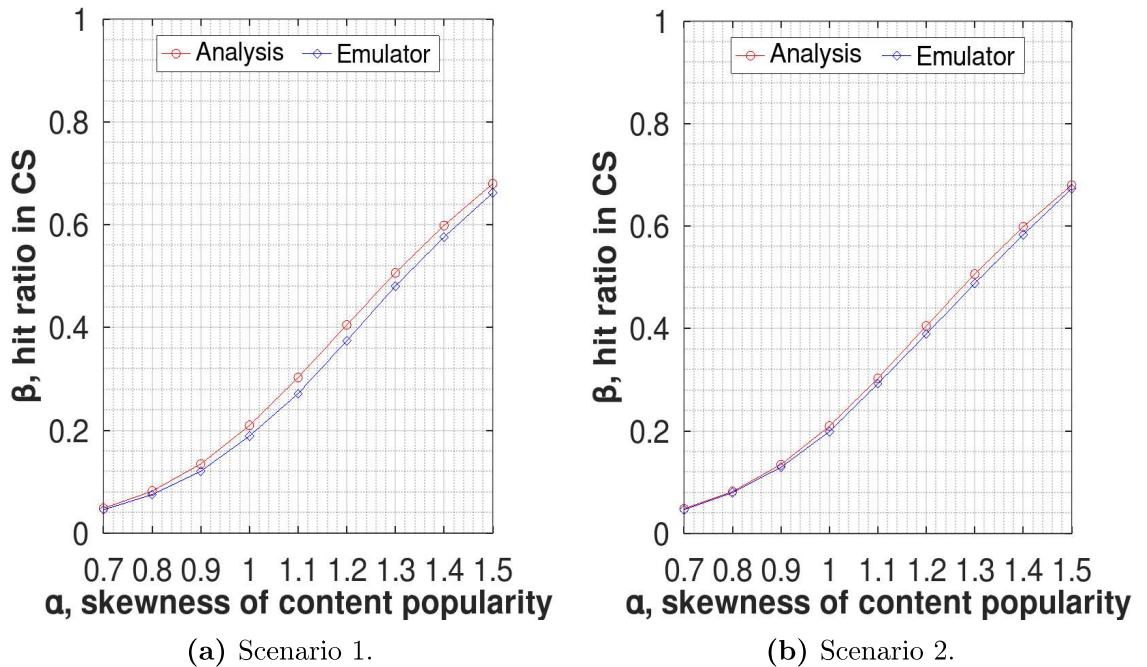
The major parameter set-up in the case of static prefix-name binding follows the parameter values as shown in Table 6.1.

**Table 5.1:** Setting values of main parameters in static prefix-name binding scheme.

Parameter	Value
Number of content items ( $M$ )	1000
Cache-size ( $C$ )	10, 100
Skewness of content popularity ( $\alpha$ )	0.7 - 1.5
Interest rate	10000 interests/s

### 5.1.1 Hit ratio

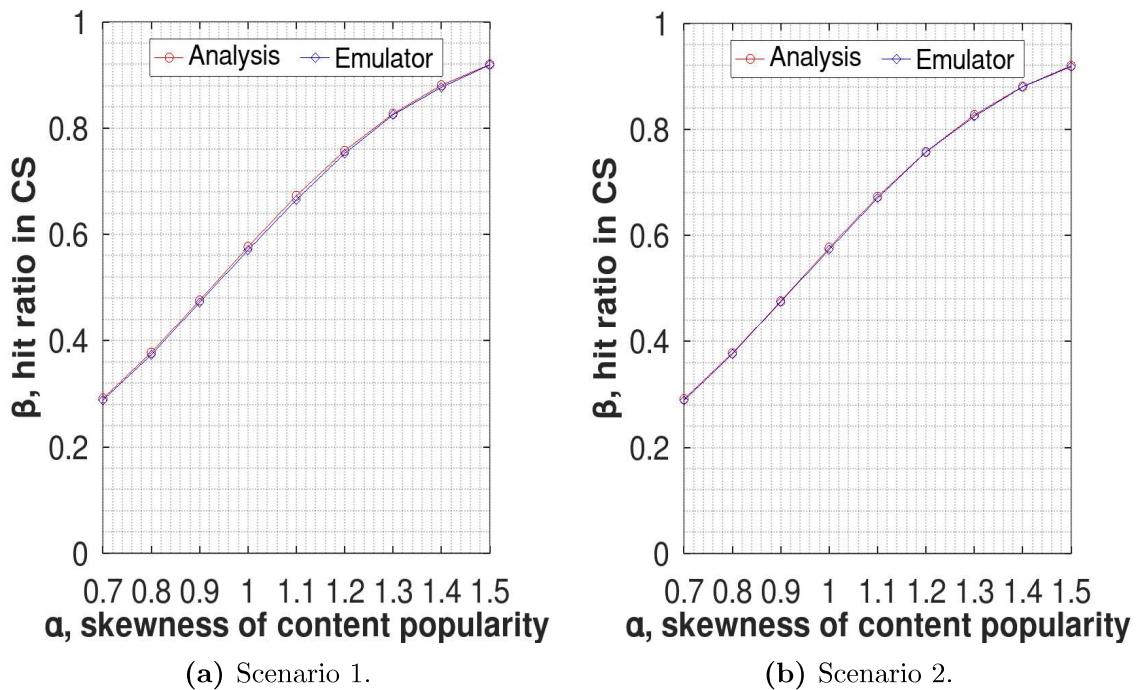
The emulation testbed values of the CS hit ratio,  $\beta$ , were measured and compared against the analysis model in both scenarios, and there were also different CS sizes.



**Figure 5.1:** Comparison of hit ratios of CS when CS capacity is equal to 1% from total number of prefix names in the case of static prefix-name binding scheme.

Figure 5-1 shows a comparison of CS hit ratios between the analysis and emulation testbeds when the CS size was equal to 10 content items. Alternatively, Figure 5-2 illustrates a comparison of CS hit ratios between the analysis and emulation testbeds when the CS size is equal to 10 content items. The analysis hit ratio,  $\beta$ , is obtained from equations (4.5) and (4.6).

The hit ratio of CS,  $\beta$ , was constantly increased from 5% to 68% when the content request skewness,  $\alpha$  increased from 0.7 to 1.5 for CS size equal to 10 out of 1000 content items in both scenarios 1 and 2. The gap between the analytical and emulation CS hit ratios,  $\Delta\beta$ , was about 5% on average, where the gap monotonously increased when  $\alpha$  increased for scenario 1. In the case of scenario 2, the gap of  $\beta$  also followed the trend of scenario 1 with a gap about 2% smaller than scenario 1 as shown in Figure 5-1.



**Figure 5-2:** Comparison of hit ratios of CS when CS capacity is equal to 10% from total number of prefix names in the case of static prefix-name binding scheme.

Whenever the size of CS increased, i.e., CS equaled 100 content items, a significant rise occurred in the CS hit ratio,  $\beta$ , from 5% to 29% at  $\alpha = 0.7$  and 68% to 92% at  $\alpha = 1.5$ . Moreover, the gap of  $\beta$  decreased impressively to about 0.1% for all axes of

$\alpha$  in both scenarios 1 and 2 as seen in Figure 5-2. We assume that this phenomenon occurred because of delayed caching, which was revealed in [31].

### 5.1.2 Processing time distribution

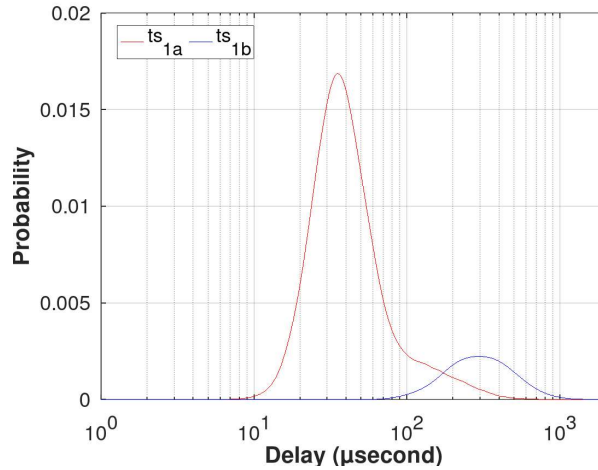
Processing time is essentially required to calculate the throughput. It is constructed from an accumulation of delay components in each case of packet flow in the translation process. Therefore, the processing time distribution can determine the range and deviation of each case of translation. We used two values, median and mean, taken from five hundred thousand packets in our emulation testbed experiment. Table 5.2 shows the median and mean value of each delay component.

**Table 5.2:** Statistical results of delay components.

Symbol	Median	Mean	Symbol	Median	Mean
$t_c$	1 $\mu s$	1.1 $\mu s$	$t_{d_p}$	45 $\mu s$	47.4 $\mu s$
$t_{cn}$	1 $\mu s$	0.9 $\mu s$	$t_{d_s}$	130 $\mu s$	141.8 $\mu s$
$t_{reg1}$	1 $\mu s$	1.2 $\mu s$	$t_{ip_c}$	4 $\mu s$	5.5 $\mu s$
$t_{reg1'}$	1 $\mu s$	1.1 $\mu s$	$t_{ip_d}$	5 $\mu s$	6.5 $\mu s$
$t_{reg2}$	1 $\mu s$	1.2 $\mu s$	$t_{pr1}$	95 $\mu s$	98.5 $\mu s$
$t_{reg2'}$	1 $\mu s$	1.1 $\mu s$	$t_{pr2}$	111 $\mu s$	115 $\mu s$
$t_{ip}$	44 $\mu s$	47 $\mu s$	$t_{soh}$	24 $\mu s$	24.7 $\mu s$
$t_{i_s}$	88 $\mu s$	90.5 $\mu s$	$t_{nr_s}$	10000 $\mu s$	10000 $\mu s$

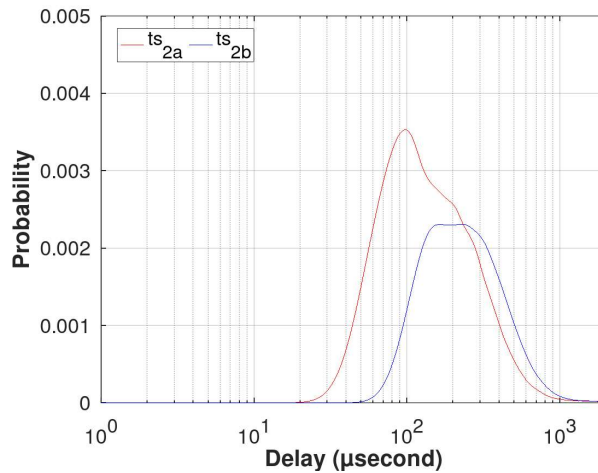
Figure 5-3 shows the distribution of processing times  $ts_{1a}$  and  $ts_{1b}$  in the case of scenario 1. In general, the distribution of  $ts_{1b}$  is wider than  $ts_{1a}$ . This is because the number of delay components in  $ts_{1b}$  was more than the number for  $ts_{1a}$ . The accumulation of several delay components resulted in a longer processing time in  $ts_{1b}$ . The distribution of  $ts_{1b}$  was shifted to the right from  $ts_{1a}$  at a scale of 100  $\mu s$ . The processing time interval of  $ts_{1a}$  reached about 1000 times between the min and max values. Moreover, the distribution skewness of  $ts_{1a}$  and  $ts_{1b}$  appeared similar, that is, skewed to the right, since the mean was slightly longer than the median of the processing time.

Figure 5-4 shows the processing time distribution of  $ts_{2a}$  and  $ts_{2b}$  in the case of scenario 2. The distribution of  $ts_{2a}$  was centered at about 80  $\mu s$  with a minimum processing time of around 10  $\mu s$  and a maximum of about 1000  $\mu s$ . With the smaller range, the interval of  $ts_{2b}$  was also similar to  $ts_{2a}$  with a starting point from 50  $\mu s$  to 1000  $\mu s$ . Moreover, its center value was shifted to the right around 170  $\mu s$  from the central value of  $ts_{2a}$ . In addition, the number of delay components for  $ts_{2b}$  was more



**Figure 5-3:** Distribution of processing time components in the case of scenario 1 and static prefix-name binding scheme.

than the number for  $ts_{2a}$ . The distribution of  $ts_{2b}$  was also affected by the producer response time, which contributed to a longer accumulation of processing time.



**Figure 5-4:** Distribution of processing time components in the case of scenario 2 and static prefix-name binding scheme.

### 5.1.3 Throughput

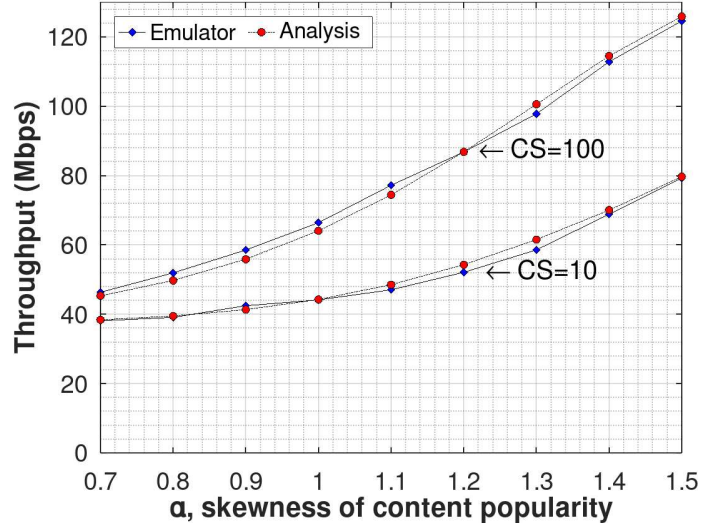
The throughput of a gateway can be determined by the number of packets sent by the gateway to a customer in a second. The emulation throughput was determined by



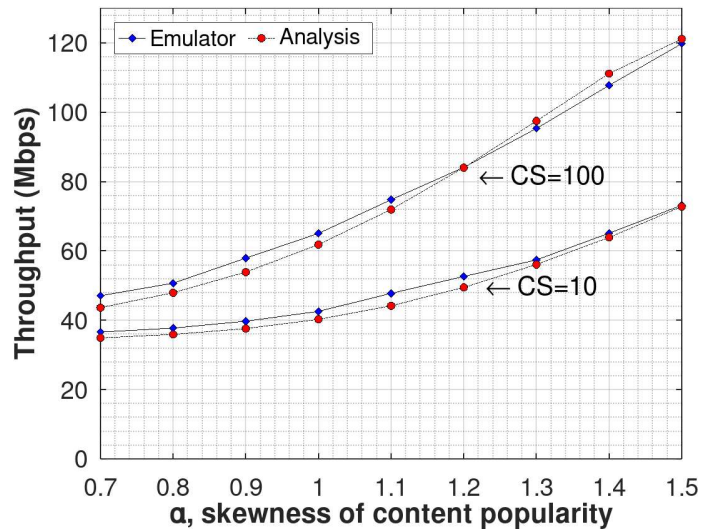
capturing each second within five minutes in the emulation testbed. Two statistical variables, namely the median and mean, were then utilized to compare the emulation and analysis throughputs. A throughput analysis was calculated by using the analytical hit ratio and the statistic of processing time taken from emulation testbed. As a result, a median and mean throughput analysis could be done with this calculation.

Figure 5-5 depicts a throughput comparison between the emulation and analysis in the case of scenario 1 by using the median and mean with a CS capacity equal to 1% and 10% of the total number of prefix names, respectively. The increase in  $\alpha$  caused a constant increase in the throughput in different cases of CS sizes. This is because the consumer requested more popular content, which lead to an increase in the CS hit ratio, so the parameter  $\beta$  made a dominant contribution to the increase in throughput. For example, when the CS size was 10 content items, the throughput increased from about 40 Mbps at  $\alpha = 0.7$  to 80 Mbps at  $\alpha = 1.5$  for the median approach. However, in the mean approach, the increase in throughput was 5% less from the median. Another fact shows that an increase in CS size of about 10 times resulted in an increase in the average throughput along the  $\alpha$  axis of about 30% on average for both the median and the mean. This is also because of the increase in the hit-ratio property  $\beta$ .

In terms of a comparison of emulation and analysis throughput, the difference between analysis and emulation for the median was generally lower than the mean. In the case of a CS size equal to 10 items, the gap between emulation and analysis throughput was about less than 5% for the median compared with 8% for the mean with a gap that was higher in the middle of the  $\alpha$  axis between 0.9 and 1.4. However, it was smaller at the edge of the  $\alpha$  axis. This is because the property of the hit ratio and processing time changed over the  $\alpha$  axis. When  $\alpha$  was at the minimum value, the hit ratio gap was lower, but it widened as  $\alpha$  increased. When  $\alpha$  was at the minimum value, the processing time was at the maximum, but it shortened as  $\alpha$  increased. Therefore, the contribution of the longer processing time and higher gap in hit ratio at the middle of the  $\alpha$  axis caused a decrease in accuracy performance. However, in the case of a CS size equal to 100 items, the gap between analysis and emulation was generally higher at a lower  $\alpha$  of about 5% and 7% in the median and mean approaches, respectively. This is because the increase in CS size affected the difference in the hit ratio between emulation and analysis, making it close to zero. Therefore, the cause of differences between the analysis and emulation was only affected by the property of processing time, where the processing time decreased when alpha increased.



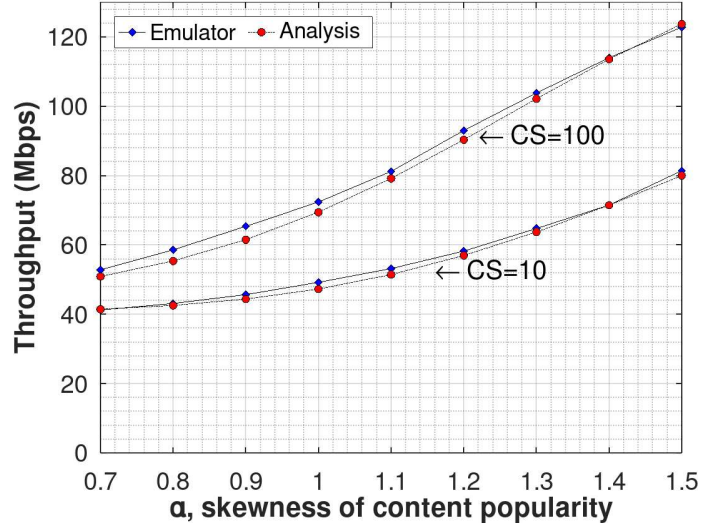
(a) Median approach.



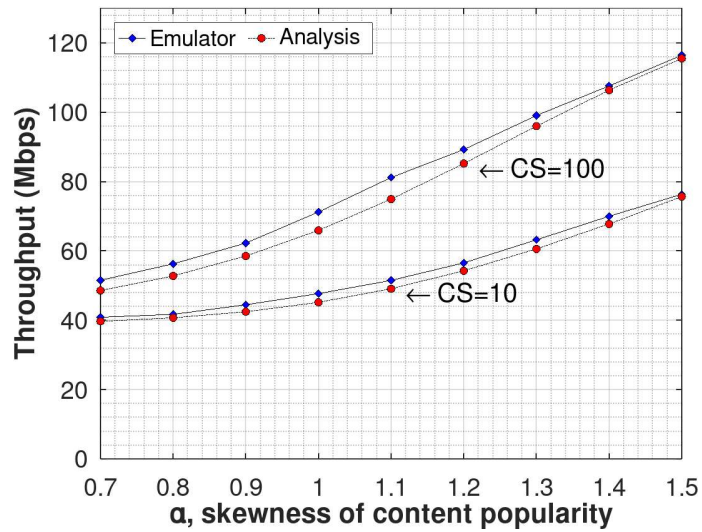
(b) Mean approach.

**Figure 5-5:** Throughput comparison between estimation analysis and emulation testbed when CS size was equal to 1% and 10% from total number of prefix names in the case of scenario 1 using static prefix-name binding scheme.

Figure 5-6 illustrates a throughput comparison between emulation and analysis in the case of scenario 2 by using the median and mean with a CS capacity equal to 1% and 10% of the total number of prefix names, respectively. Similar to scenario



(a) Median approach.



(b) Mean approach.

**Figure 5-6:** Throughput comparison between estimation analysis and emulation testbed when CS size was equal to 1% and 10% from total number of prefix names in the case of scenario 2 using static prefix-name binding scheme.

1, the increase in  $\alpha$  caused a constant increase in the throughput for both cases of CS sizes equal to 1% and 10%. The throughput increased from about 41 Mbps at  $\alpha = 0.7$  to 81 Mbps at  $\alpha = 1.5$  for the median approach. However, for the mean

approach, the throughput increased from about 38 Mbps at  $\alpha = 0.7$  to 74 Mbps at  $\alpha = 1.5$ . The increase in CS size from 10 to 100 resulted in a significant increase of about 35% of average throughput along the  $\alpha$  axis for both the median and mean. The difference between analysis and emulation for the median was also lower than the mean approach. The average deviation was about 3% for the median compared with 5% for the mean for CS sizes equal to 10 out of 1000 content items and 7% for the median compared with 10% for the mean for CS size equals to 100 out of 1000 items.

Having a similar behavior, the difference between scenarios 1 and 2 was highlighted at the cross point between analysis and emulation throughput, especially when the CS capacity was equal to 10% of the total number of prefix names. In the beginning, the analysis throughput was located below the emulation graph at an  $\alpha$  of less than 1.2, and it exceeded the emulation graph afterward in the case of scenario 1. However, in the case of scenario 2, the analysis throughput crossed the emulation graph at an  $\alpha$  of higher than 1.4. This is because scenario 1 had a shorter processing time at a higher alpha than scenario 2. As a result, it improved the overall analysis throughput estimation.

## 5.2 Dynamic prefix-name binding

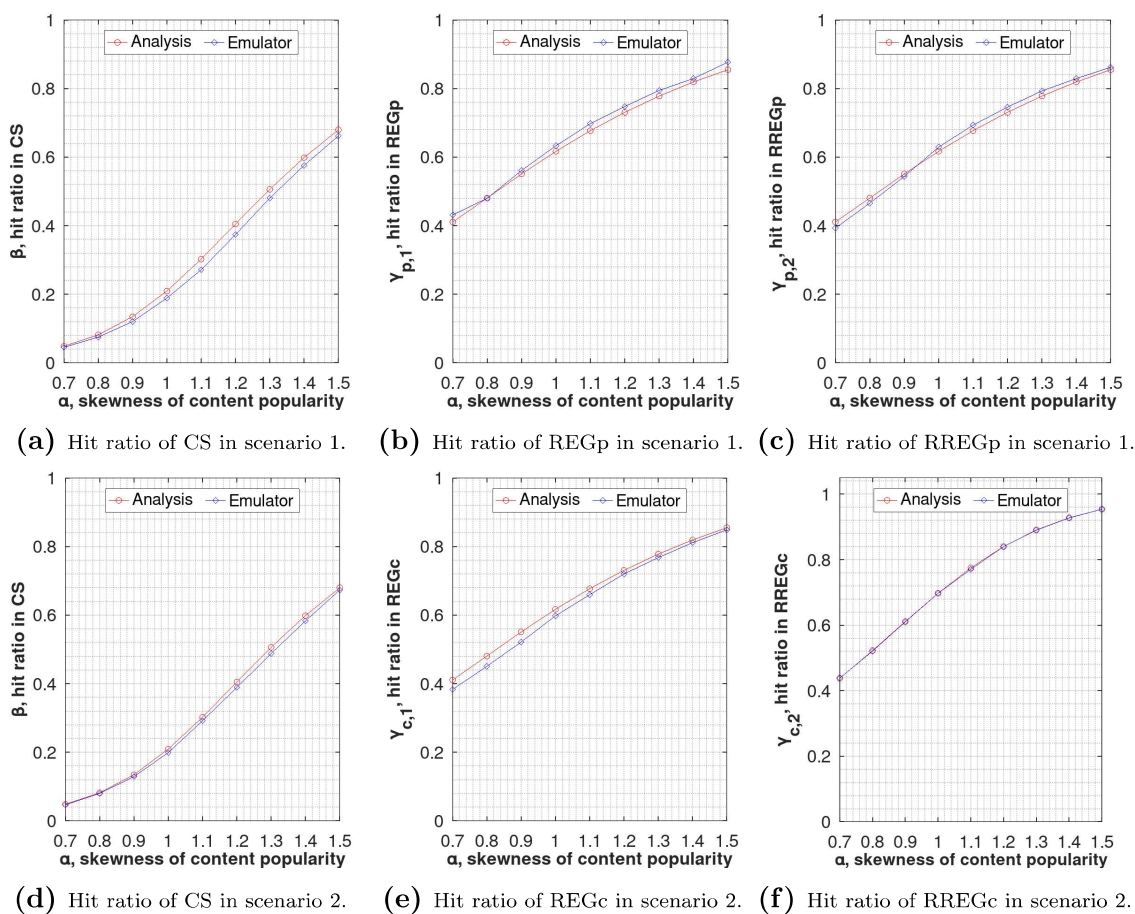
The major parameter set-up in the case of dynamic prefix-name binding followed the parameter values shown in Table 5.3. In this case, the dual-channel gateway only caches a limited number of prefix names due to its limited memory capacity. The NRS that holds all associations of prefix name and IP address will immediately send a response when a request is sent by the gateway. The number of prefix names cached in the REG table should be far bigger than the number of unique content items in CS. This is because the REG table only stores a prefix name and its associated IP address, which requires less memory compared with the data content.

### 5.2.1 Hit ratio

The emulation values of  $\beta$ ,  $\gamma_{p,1}$ ,  $\gamma_{p,2}$ ,  $\gamma_{c,1}$ , and  $\gamma_{c,2}$  were measured and compared against the analysis model seen in Figure 5.7. In the case of scenario 1, the hit ratio of CS,  $\beta$ , constantly increased from 5% to 68% when the content request skewness,  $\alpha$ , increased from 0.7 to 1.5. Moreover, the gap of  $\beta$  between the analytical model and emulation increased as  $\alpha$  increased by up to 5% as seen in Figure 5.7(a). As for

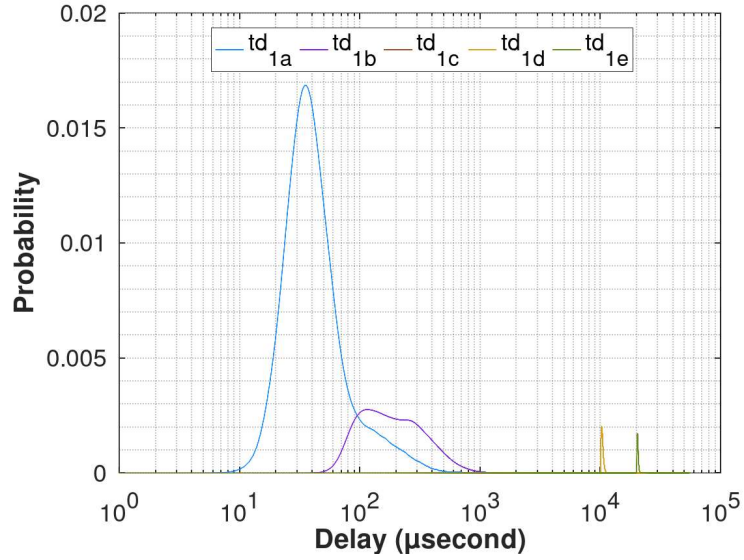
**Table 5.3:** Setting value of main parameters in dynamic prefix-name binding scheme.

Parameter	Value
Number of content items ( $M$ )	1000
Cache size ( $C$ )	10
Reg size ( $R$ )	200
Skewness of content popularity ( $\alpha$ )	0.7 - 1.5
Interest rate	10000 interests/s

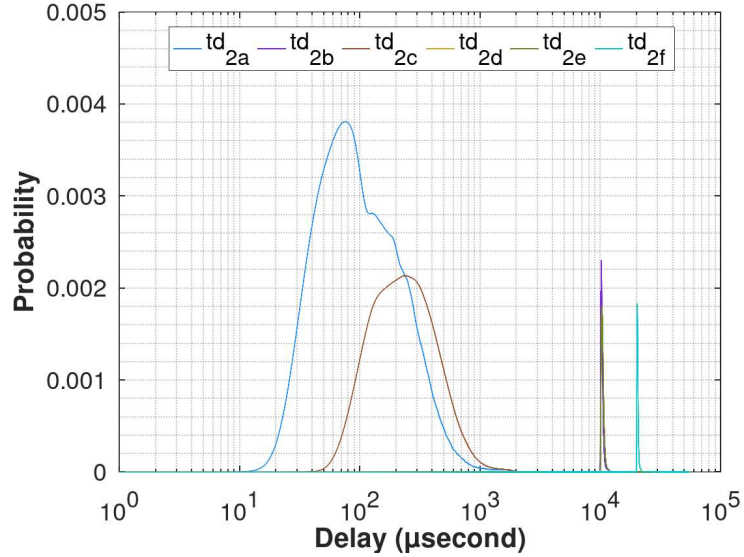


**Figure 5.7:** Comparison of hit ratio for CS ( $\beta$ ), REG producer ( $\gamma_{p,1}$ ), RREG producer ( $\gamma_{p,2}$ ), REG consumer ( $\gamma_{c,1}$ ), and RREG consumer ( $\gamma_{c,2}$ ) when CS and REG size equal 1% and 20%, respectively, of total number of prefix names in the case of dynamic prefix-name binding.

$\gamma_{p,1}$  and  $\gamma_{p,2}$  with an REG size of 200 out of 1000 items, the hit ratios,  $\gamma_{p,1}$  and  $\gamma_{p,2}$ , gradually increased from 41% to 85% when the content request skewness,  $\alpha$ , increased



**Figure 5-8:** Distribution of processing time components in the case of scenario 1 and dynamic prefix-name binding scheme.



**Figure 5-9:** Distribution of processing time components in the case of scenario 2 and dynamic prefix-name binding scheme.

from 0.7 to 1.5. Furthermore, the gap hit ratios of  $\gamma_{p,1}$  and  $\gamma_{p,2}$  were quite similar to each other with an average of 2% oscillation along with the  $\alpha$  axis as shown in Figure 5-7(b) and (c).

In the case of scenario 2, the hit ratio of CS,  $\beta$ , was also followed a similar trend as scenario 1 with a rise from 5% to 68% when the content request skewness,  $\alpha$ , increased from 0.7 to 1.5. The gap of  $\beta$  between the analysis and emulation was 2% smaller than in scenario 1 as shown in Figure 5-7(d). The  $\gamma_{c,1}$  and  $\gamma_{c,2}$  behaved differently due to different ratios of content request. The  $\gamma_{c,1}$  request ratio came from  $q(i)$ , while that of  $\gamma_{c,2}$  came from  $r(i)$ . As a result,  $\gamma_{c,1}$  increased from 41% to 85% as shown in Figure 5-7(e), while  $\gamma_{c,2}$  increased from 43% to 95% as seen in Figure 5-7(f) when the content request skewness,  $\alpha$ , developed from 0.7 to 1.5 with an REG size equal to 20% of the total number of prefix names. Furthermore, the  $\gamma_{c,1}$  gap began at around 3% and decreased as  $\alpha$  increased. However, the gap between  $\gamma_{c,2}$  was nearly zero for all values of  $\alpha$ .

### 5.2.2 Processing time distribution

In the case of dynamic prefix-name binding deployment, scenario 1 had five different processing time distributions, whereas scenario 2 had six. The introductions of the hit ratio of the REG<sub>p</sub>, REG<sub>c</sub>, RREG<sub>p</sub>, and RREG<sub>c</sub> were the main cause for the increase in processing time. Moreover, the NRS resolving round trip time also dominantly affected some of the distributions, causing them to shift to the far right in processing time distribution.

Figure 5-8 illustrates the distribution in processing time for various types of packet flows in scenario 1. There were five distinct instances of packet processing time from  $td_{1a}$  to  $td_{1e}$ . The  $td_{1a}$  distribution had the shortest processing time with an interval between 10 and 900  $\mu$ s, whereas the  $td_{1b}$  distribution had the second shortest processing time between 80 and 1000  $\mu$ s, followed by the  $td_{1c}$  and  $td_{1d}$  distributions, which overlapped each other with a central value of around 10 milliseconds and an interval between maximum and minimum of about 1000  $\mu$ s. This overlap was caused by these distributions accessing the NRS server once. Finally,  $td_{1e}$  was the longest with a central value of about 20 milliseconds since  $td_{1e}$  accessed the NRS server twice.

In the case of scenario 2, there were six distinct cases of packet-processing time distribution, with  $td_{2a}$  being the smallest followed by  $td_{2c}$ ,  $td_{2b}$ ,  $td_{2d}$ ,  $td_{2e}$ , and  $td_{2f}$  as seen in Figure 5-9. The distribution of  $td_{2a}$  was in the approximate range of 10 to 900  $\mu$ s, whereas the value of  $td_{2c}$  was in the range of 80 to 1000  $\mu$ s. Interestingly, the distributions of  $td_{2b}$ ,  $td_{2d}$ , and  $td_{2e}$  mainly overlapped with each other, with a median of roughly 10 ms. As expected, this is because the distribution of  $td_{2b}$ ,  $td_{2d}$ , and  $td_{2e}$  experienced the NRS resolving round trip time once. As the longest, the distribution

of  $td_{2f}$  had a center processing time of roughly 20 milliseconds because it experienced the NRS resolving round trip time twice.

### 5.2.3 Throughput

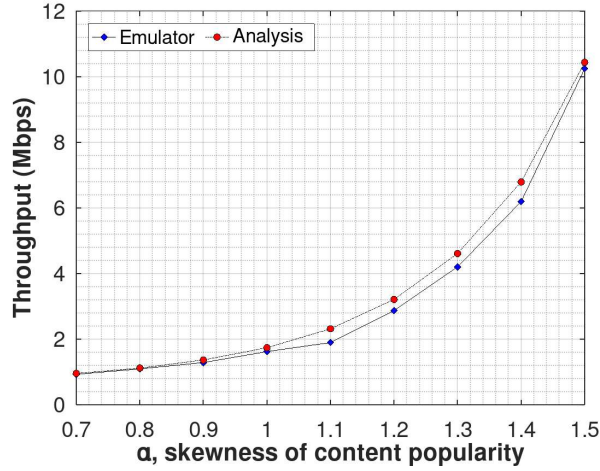
Figure 5-10 shows the throughput comparison between emulation and analysis with the two different statistic approaches, median and mean, in the case of scenario 1. Alternatively, Figure 5-11 depicts the comparison of the throughput in the case of scenario 2.

With the capacity, only 20% of the total number of prefix names were stored in the REG table. The throughput of the dual-channel gateway in dynamic prefix-name binding was between 2.5% and 25% from the static prefix-name binding along the  $\alpha$  axis when the CS capacity was equal to 1% of the total number of prefix names. Additional requests to the NRS server caused a significant drop in the overall throughput of the dual-channel gateway because they led to a longer processing time.

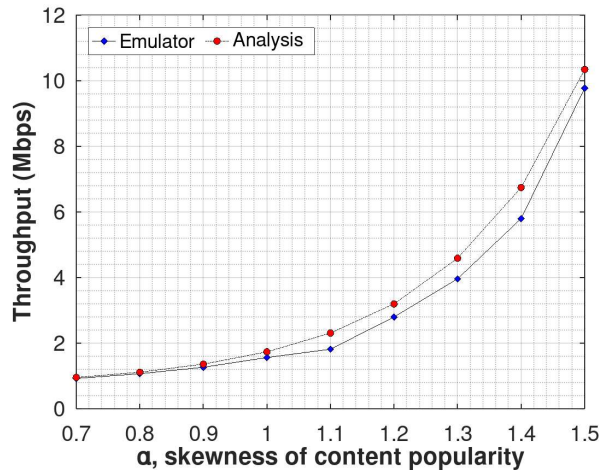
In dynamic prefix-name binding, similar to static prefix-name binding, the median approach had a higher accuracy than the mean for both scenarios. In the case of scenario 1, the throughput gap between analysis and emulation was less than 10%, with the median being slightly better than the mean at less than 15% as seen in Figure 5-10. Moreover, scenario 2 had a higher accuracy compared with scenario 1. The analysis model could estimate throughput with a 95% accuracy when using the median approach but with only an 85% accuracy when using the mean approach, as shown in Figure 5-11. This is because scenario 2 had a smaller difference in the hit ratio comparison, particularly for  $\beta$ , than scenario 1. In addition to that, the hit ratio gap between analysis and emulation in the REG consumer table, especially  $\gamma_{c,2}$ , was close to zero for any value of  $\alpha$ . As a result, the estimation accuracy in the case of scenario 2 slightly improved.

The estimation throughput was close to that for emulation when the value of  $\alpha$  was either extremely low or high. As for a low  $\alpha$ , the throughput could be accurately estimated because the gap of the CS hit ratio,  $\beta$ , between the analysis and simulation, was near zero; then it increased constantly as  $\alpha$  increased. However, for a high  $\alpha$ , the contribution of shorter processing times, such as  $td_{1a}$  and  $td_{2a}$ , improved the accuracy of estimation since the values of  $td_{1a}$  and  $td_{2a}$  had a lower number of delay components. A number of delay components in packet processing results in a longer processing time. Therefore, when the value of  $\alpha$  was located in the middle of the graph, the accuracy of throughput estimation underperformed. This was because





(a) Median approach.

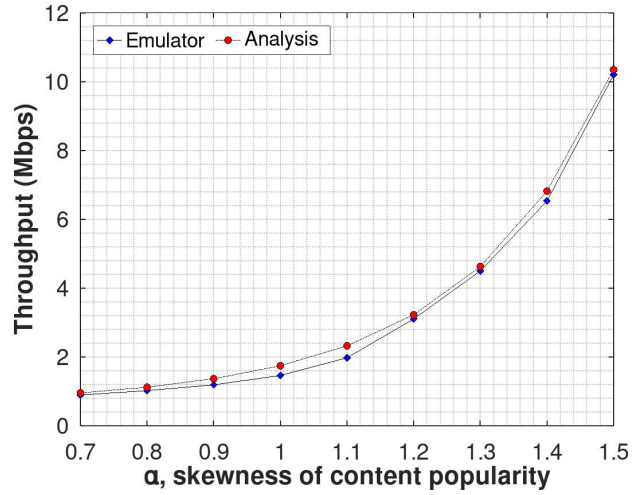


(b) Mean approach.

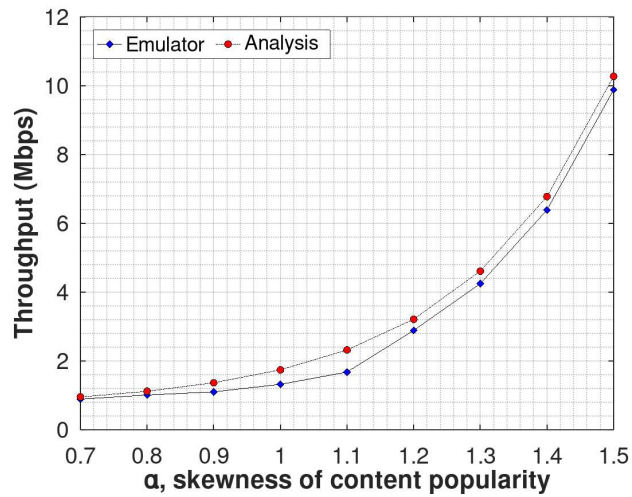
**Figure 5-10:** Throughput comparison between analysis and emulation when CS size was 1% of total number of prefix names in the case of scenario 1 and dynamic prefix-name binding scheme.

of the contribution of hit-ratio inaccuracy and longer packet processing time, which had created poor throughput estimation. This explains why a higher deviation was located in the middle of the  $\alpha$  axis.

We investigated the inaccuracy of the hit-ratio phenomenon between analysis and emulation shown in Figure 5-7. We assume that this phenomenon is closely related to delayed hit caching reported in [31]. Thus, we analyze the effect of delayed hit caching in our translation gateway. Furthermore, we propose also a method to suppress the



(a) Median approach.



(b) Mean approach.

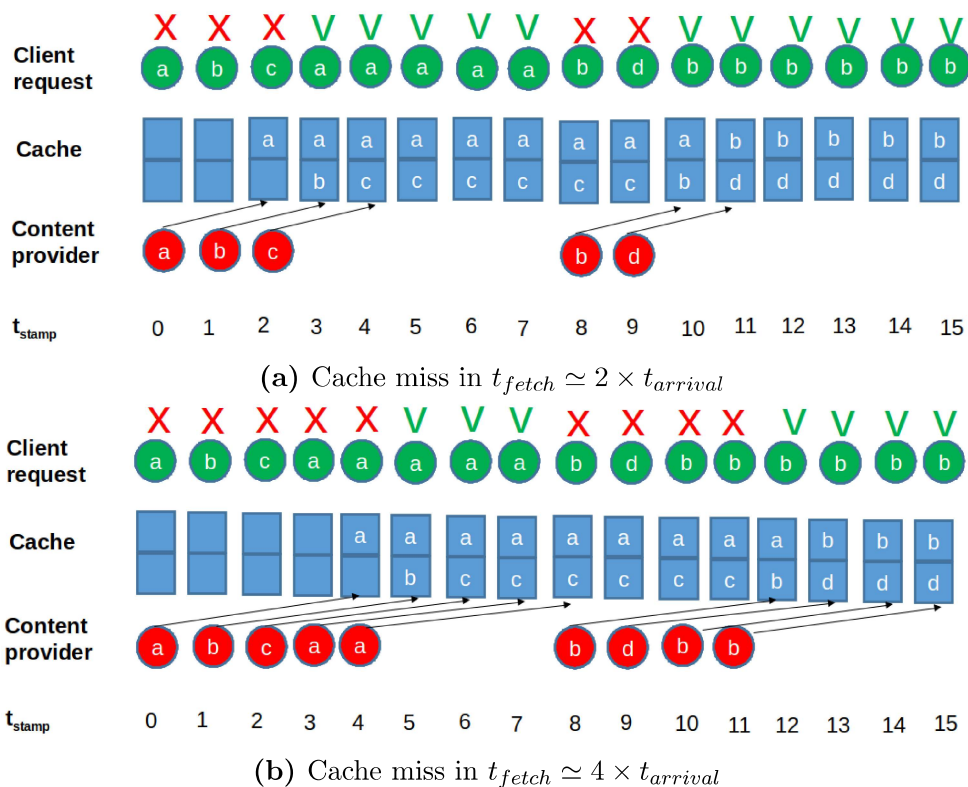
**Figure 5-11:** Throughput comparison between analysis and emulation when CS size was 1% of total number of prefix names in the case of scenario 2 and dynamic prefix-name binding scheme.

impact of delayed hit caching as described in chapter 6.

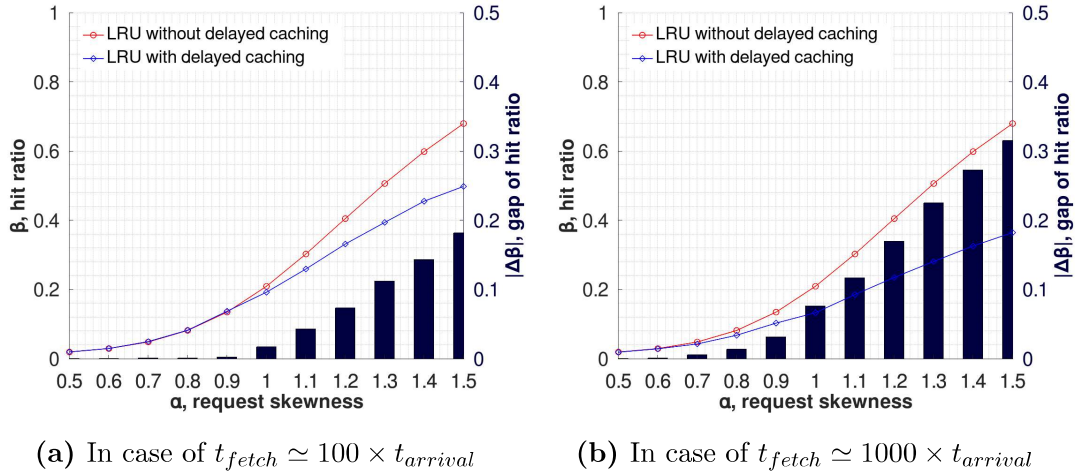
# Chapter 6

## Delayed Caching

This chapter describes the effect of delayed hit caching or delayed caching in an online-caching system, particularly in a translation gateway. To give a better understanding, Figure 6-1 depicts a typical cache with a delay caching that occurred in the caching system. Whenever a client requests content but the content absences in the cache, the proxy or gateway forwards this request packet to the original content provider to fetch the data content. The retrieval process takes some time, known as



**Figure 6-1:** Cache miss because of delayed caching in different length of  $t_{fetch}$  relative to  $t_{arrival}$



**Figure 6.2:** The hit-ratio ( $\beta$ ) decline in LRU in different value of  $t_{fetch}$  relative to  $t_{arrival}$ .

the producer response time ( $t_{fetch}$ ). And the average arrival time between the client's requested packets, known as  $t_{arrival}$ . If a new request for the same data content arrives at the cache before  $t_{fetch}$  is completed, this new request suffers from a cache miss in the cache. Then, if the condition repeatedly occurs in the online-caching server, it causes a significant hit ratio decline.

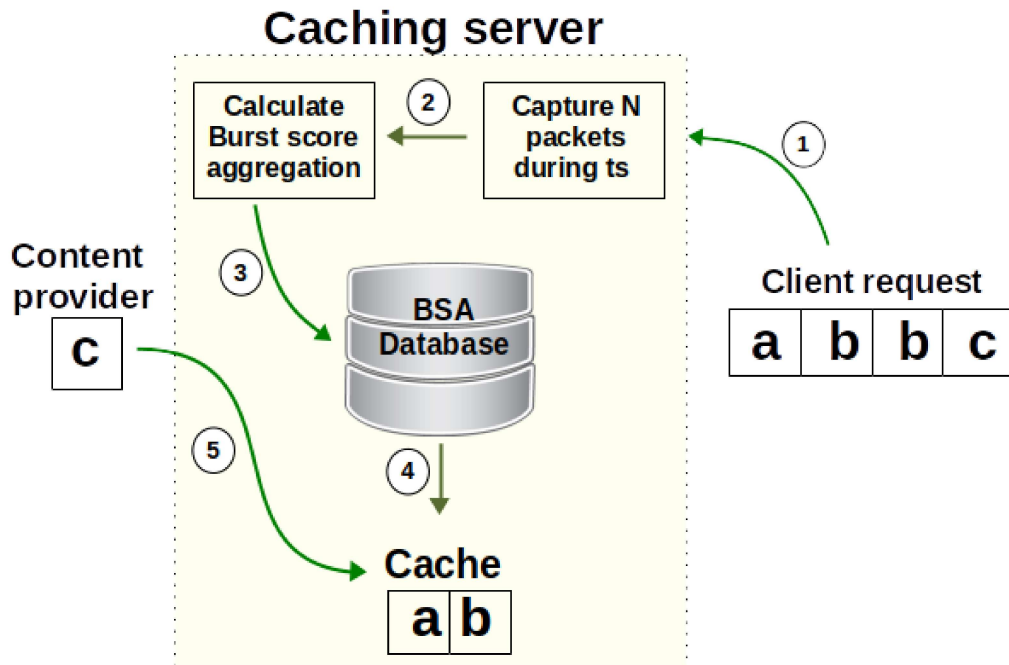
The hit ratio is defined as the number of requests satisfied by the cache divided by the number of all requests that arrived at the cache. The higher the hit ratio is or the lower the miss ratio is, the more requests are satisfied by the cache. The hit-ratio degradation that occurs due to delay caching is depicted in Figure 6.1. Figure 6.1(a) shows the case when  $t_{fetch}$  is two times of  $t_{arrival}$ , whereas Figure 6.1(b) shows the case when  $t_{fetch}$  is four times of  $t_{arrival}$ . In the first case, just eleven requests are satisfied by the cache, whereas seven requests are satisfied by the cache in the second case. The longer  $t_{fetch}$  relative to  $t_{arrival}$  can cause more cache miss. As a result, the hit ratio is declining in the online-caching server.

Figure 6.2 shows the hit-ratio degradation that exist in online-caching server. The hit-ratio degradation is subjected to the incoming request that obeys Zipf distribution representing the content popularity and the comparison between  $t_{fetch}$  and  $t_{arrival}$ . It shows that the smaller  $\alpha$  creates smaller gap compare to higher  $\alpha$  in every different case of  $t_{fetch}$  relative to  $t_{arrival}$ . Furthermore,  $\Delta\beta$ , the gap between idealized LRU and delayed caching LRU, monotonically increased as  $t_{fetch}$  increased. The average of  $\Delta\beta$  were about 15% and 30% when  $t_{fetch} \simeq 100 \times t_{arrival}$ , and  $t_{fetch} \simeq 1000 \times t_{arrival}$

respectively. We have found that the more popular content was higher likely to experience delayed caching. This is because more requests for popular content were more likely to be requested during the  $t_{fetch}$  interval, so they experienced cache miss in the cache.

## 6.1 Delayed caching suppressing method

A specific strategy to suppress the effect of delayed caching is developed. The technique relies on the burstiness of contents as a key parameter to prioritize the residency in the cache system. The procedure of the proposed method can be generally described in Figure 6-3. First, we need to capture  $L$  number of requests during fetching time  $t_{fetch}$ . Next, a burst score is calculated and aggregated from the  $L$  number content's request. A database consisting of records from all the BSA (Burst Score Aggregation) of unique contents is built. Finally, the cache implements a cache replacement algorithm that evicts the content with the lowest BSA whenever the capacity is full after receiving content from the producer. Therefore, the parameter burst score and BSA are the essential keys in this suppressing method.



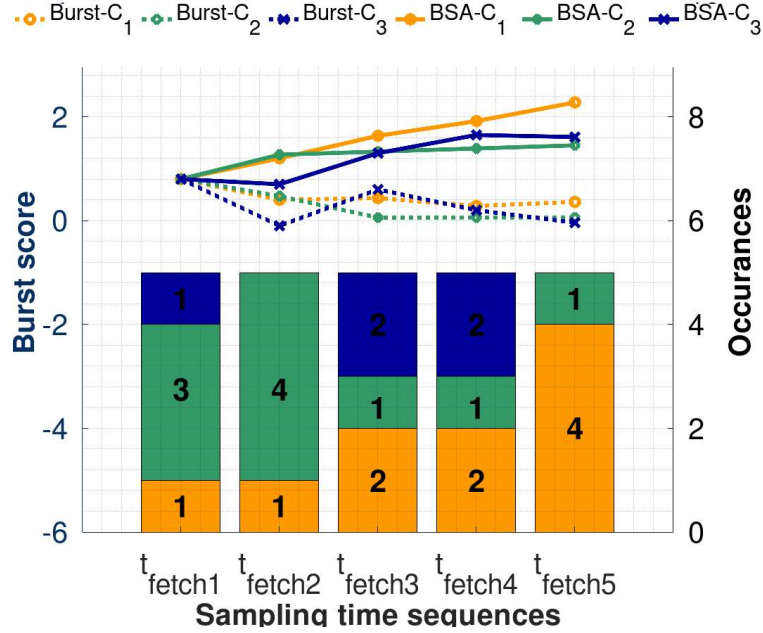
**Figure 6-3:** Architecture of BSA based cache replacement algorithm

### 6.1.1 Burst score

We have shown that the popular content has a higher probable experiencing delayed caching in the previous section. Under certain conditions, a particular content occurrence is dominant at a specific sampling time compared to other sampling times. This situation is defined as a burst period. The burst score of the particular content can be measured by comparing the occurrences of that content between sampling time and elapsed time. Hoonlor et al. in [2] introduced the burst score of a particular content  $x$  for the specific sampling period, which is similar to  $t_{fetch}$  in an online-caching server, which can be defined by

$$Burst(x, t_{fetch}) = \left( \frac{E_{t_{fetch}}}{E} - \frac{1}{T} \right), \quad (6.1)$$

where  $E_{t_{fetch}}$  is the total number of occurrences of event  $x \in \{c_1, \dots, c_N\}$  in sampling time interval of  $t_{fetch}$  and  $E$  is the total number of occurrences of  $x \in \{c_1, \dots, c_N\}$  in total elapse time  $T$ . The set of  $\{c_1, \dots, c_N\}$  express the number of  $N$  total unique contents.



**Figure 6-4:** Burst score and Burst score aggregation (BSA) with five sampling period,  $t_{fetch}$ , and three unique contents.

As for example, let's consider five sampling periods namely  $t_{fetch1}$ ,  $t_{fetch2}$ ,  $t_{fetch3}$ ,  $t_{fetch4}$ , and  $t_{fetch5}$  respectively, with three different contents  $c_1$ ,  $c_2$ , and  $c_3$  as seen in Figure 6.4. The average number of content in each sampling time is five contents. This value can be obtained by dividing  $t_{fetch}$  with  $t_{arrival}$ . Using formula 6.1, it is obtained that the content's burst score is at maximum when the content appears for the first time during the sampling period. Afterwards, it is about in the range between zero and the maximum value, or always in positive value. On the other hand, the burst score becomes negative if the content does not appear at any rate in the sampling period. Furthermore, the burst score produces identical scores for contents that have a different number of occurrences within the sampling period and are present for the first time. As a result, the burst score can not describe the magnitude of occurrence of a specific content compared to others. It is a metric to identify a particular content's occurrence in the sampling period relative to the total of previous occurrences.

We could interpret the burstiness of a certain content when we track the burst score from the beginning sampling period up to the latest. Therefore, we introduce burst score aggregation (BSA) to measure the burst level of content by accumulating from previous burst scores up to elapse time. The BSA of the specific content can be defined by

$$BSA(x, t_{fetch}) = \sum_{i=1}^M Burst(x, t_{fetch}), \quad (6.2)$$

where  $M$  is the total number of  $t_{fetch}$  sequence up to elapse time  $T$ . Thus,  $M$  is ratio between  $T$  and  $t_{fetch}$ .

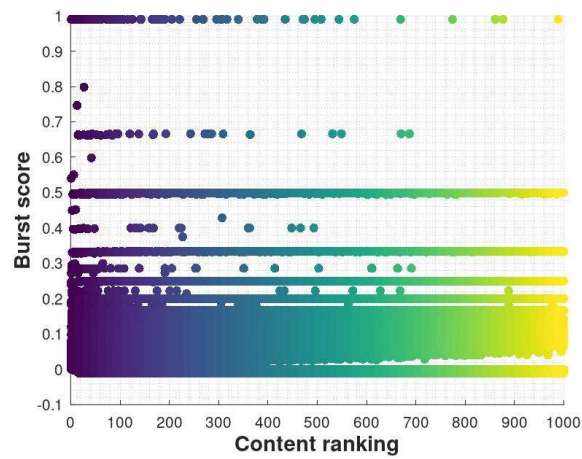
BSA necessities the accumulation of the previous burst score up to elapse time to interpret the trend of specific content being frequently requested or not. If the frequency of specific content gradually increases over time and has never been absent in all sampling period sequences, then the BSA level will reach the maximum value. Using previous example as shown in Figure 6.4, BSA of  $c_1$  has the highest value at  $t_{fetch5}$  because  $c_1$  consistently presents in all sampling period during the elapse time. Moreover, the number of occurrences is also gradually increasing. On the contrary, even though  $c_2$  has the same BSA level at the beginning of sampling time, it remains stagnant because its frequency is in a declining trend.

On the other hand,  $c_3$  becomes the second highest at  $t_{fetch5}$  even though it has a negative burst score in the beginning, but it is compensated with the next positive frequency trends. As a result, the BSA is significantly improved at  $t_{fetch5}$ . Therefore,

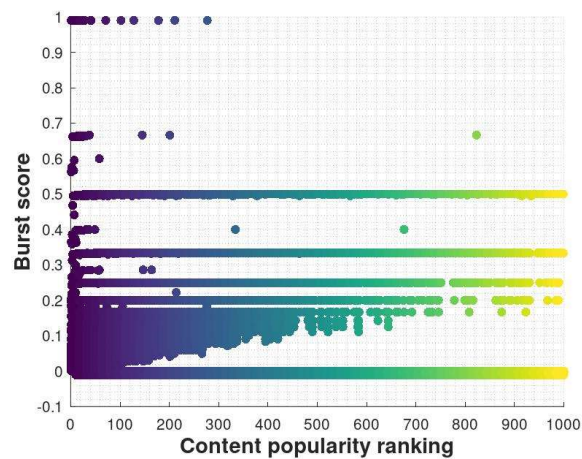


the high BSA means that the content has a high probable being dominantly requested in the future, so this parameter can be used to determine which content should be prioritized in the cache.

To understand the characteristic of BSA against request distribution, Figure 6-5 shows the burst score distribution of content that follows the Zipf distribution. Two skewness parameters of Zipf distribution for  $\alpha$  equal to 1.0 and 1.5 are taken as a comparison. The total elapsed time is about 10 seconds with a sampling interval,



(a) When  $\alpha=1.0$



(b) When  $\alpha=1.5$

**Figure 6.5:** Burst score distribution against skewness parameter ( $\alpha$ ) of Zipf distribution for 1000 unique contents ( $N$ )



$t_{fetch}$ , of about 10 ms. From the figure 6-5, we know that popular contents have a more positive burst score than unpopular content. As the value of  $\alpha$  increases, the number of positive burst scores is concentrated on the most popular content. According to Zipf distribution characteristics, the most popular content is more likely to appear more frequently. Thus, it is always found in every sequence of the sampling period.

### 6.1.2 Cache replacement algorithm

As we have mentioned earlier, the cache replacement algorithm is essential in optimizing the cache hit ratio in the caching system. We know that a cache miss in delayed caching is mainly caused by the massive arrival of the same particular content in the unfinished fetching process. Therefore, modifying the cache replacement algorithm can effectively prevent the hit-ratio decline. If a caching system prioritizes content based on its burst score aggregation, i.e., content with the lowest burst score aggregations will be evicted from the cache system. The most requested content in every sampling period will always remain in the cache. To implement this idea, the CDN server or proxy must first construct a database that records the BSA of each unique content. Second, the cache replacement policy of caching system evicts content with the lowest content burst score aggregation, as seen in Figure 6-5.

---

**Algorithm 5** Burst score database.

---

- 1:  $ts \leftarrow$  Average content provider response time;
  - 2:  $T \leftarrow$  Elapse time;
  - 3: Burst database  $\leftarrow$  Burst( $x, t_{fetch}$ )=0  $\forall x \in \{c_1, \dots, c_N\}$ ;
  - 4: **while**  $T \bmod t_{fetch} = 0$  **do**
  - 5:   Count interest Burst( $x, t_{fetch}$ )  $\forall x \in \{c_1, \dots, c_N\}$ ;
  - 6:   Aggregate burst score = Burst( $x, t_{fetch}$ )[previous]+Burst( $x, ts$ )[current]  $\forall x \in \{c_1, \dots, c_N\}$  and store in Burst database;
  - 7: **end while**
- 

Algorithm 5 shows the pseudo-code in calculating the burst scores for all available contents. Initially, the online-caching server set the value of  $t_{fetch}$  with the average producer response time. Afterward, the router calculates all burst scores in every  $t_{fetch}$  interval. The calculation result is aggregated by accumulating the previous burst score with the current value for all content items. Then, it is stored in the database.

Algorithm 6 describes the pseudo-code of content replacement in the caching server. When it receives content from a original content provider, it stores it in

---

**Algorithm 6** Content replacement

---

```
1: while Cache receives content do
2:   if Cache size is full then
3:     Evict content in cache with lowest BSA;
4:     Store content in cache;
5:   else
6:     Store content in cache;
7:   end if
8: end while
```

---

cache. However, if the capacity of cache is oversized, the online-caching server executes a content replacement algorithm that removes a particular content with the lowest BSA.

## 6.2 Evaluation result

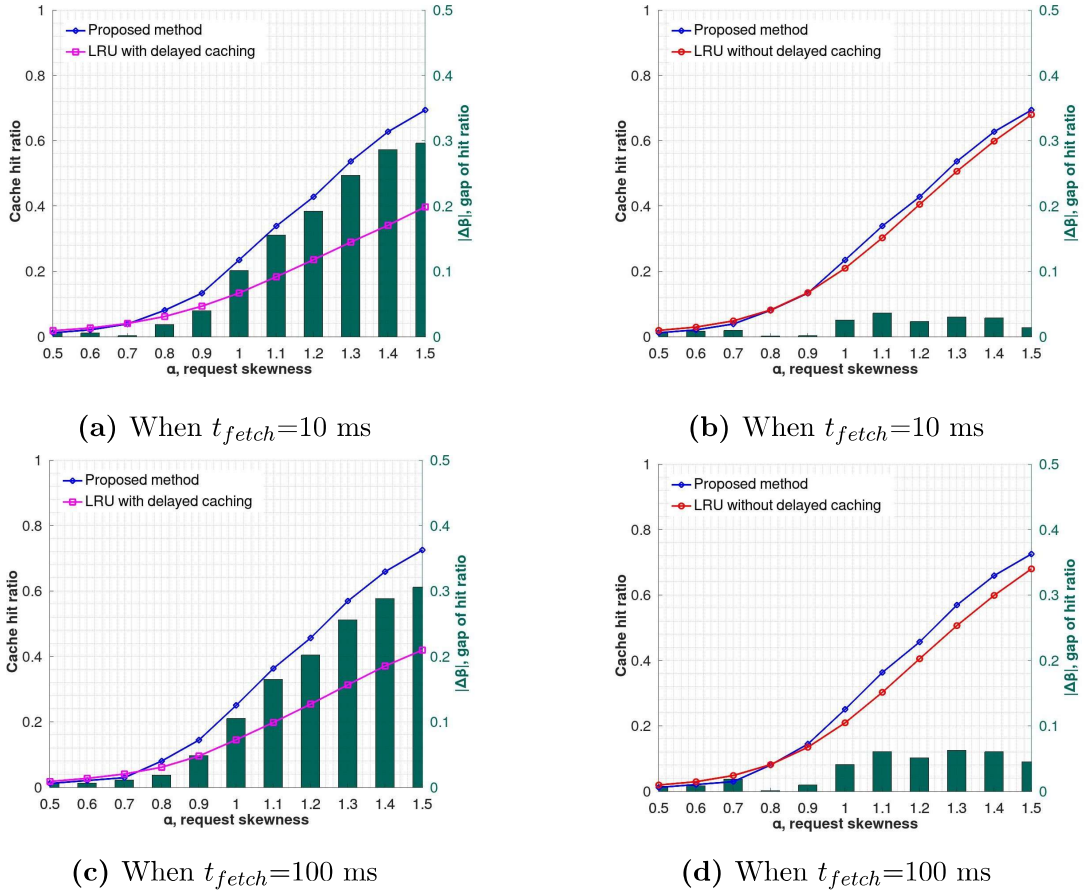
We evaluate and clarify our hypothesis using computer simulation consisting of 3 parallel processes programmed by Python 3.8. A process acts as online caching server, the others act as producer and consumer. A time delay function is equipped at producer side to provide delay effects to content fetching by caching server. Furthermore, We compare the hit ratio performance between the proposed method and LRU in the online-caching server. The sampling delay,  $t_{fetch}$ , was set to different scenarios. Table 6.1 shows the metric parameters used in the experiment.

**Table 6.1:** Setting values of main parameters

Parameter	Value
Number of unique content items ( $N$ )	1000
Cache size	10
Request skewness ( $\alpha$ )	0.5 - 1.5
Interval( $ts$ )	10 ms and 100 ms
Interest rate	10000 interests/s

We collected the data from 100000 requests sent by the requester or client. The cache hit ratios obtained by the simulator were plotted against the request skewness,  $\alpha$ , for each of the three different cases, namely LRU without delayed caching, LRU with delayed caching, and the proposed method as seen in Figure 6-6. In general, the result shows that the proposed method enable to prevent the cache hit ratio decline averagely 30% from LRU with delayed caching in case of  $t_{fetch}$  equal to 10 ms as seen in Figure 6-6(a), and 40% in case of  $t_{fetch}$  equal to 100 ms as seen in Figure 6-6(c).

The hit ratio of the proposed method outperformed the LRU without delayed



**Figure 6-6:** Comparison of hit ratio between LRU without delayed caching, proposed method, and LRU

caching, about 1% in case of  $t_{fetch}$  equal to 10 ms as shown in Figure 6-6(c) and 3% in case of  $t_{fetch}$  equal to 100 ms respectively as shown in Figure 6-6(d). This is because the proposed method utilizes the burst scores that are highly correlated with content popularity to evict the content in the cache, so the order of arrival request sequence does not affect the eviction in the cache. In addition, a high  $\alpha$  indicates that a few numbers of popular content dominate requests. The overall cache-hit ratio increases significantly when cache capacity is merely sufficient to store those popular contents. On the other hand, the LRU algorithm caches the content in accordance with the order of arrival request sequences, i.e., if the unpopular content is sent between popular content, then LRU will cache it for sure since it is the most recent. As a result, it creates more cache misses.

## Chapter 7

# Conclusions and Future work

### 7.1 Conclusion

This dissertation develops migration technology called the dual-channel IP-to-NDN translation gateway for cross-platform communication between TCP/IP and NDN protocol during the transition period that demands a cost-effective solution. The gateway uses two different channels for recognizing two types of NDN packets in the IP network. The proposed method can leverage IP protocol semantics, especially in the network layer, without exposing prefix name data privacy. A packet naming mechanism was solved by providing two asymmetric REG tables, namely REG producer and REG consumer tables, that can be connected to global name services. These tables are utilized by either the IP as a producer or consumer. We also provide analytical models of the throughput of the proposed gateway, and we showed that the proposed model could accurately estimate the throughput of the gateway.

The design and principle of the dual-channel IP-to-NDN translation gateway have been presented in chapter 3. The proposed gateway envisions the separation of interest and data packets by mapping two dedicated IP addresses as the interest channel and data channel, respectively. Two asymmetric REG tables, REG producer and REG consumer tables, entitle the name to an IP packet binding mechanism that is set statically or dynamically manner. Results are promising, and they show that:

- The dual-channel IP-to-NDN translation gateway can leverage TCP/IP into ICN/NDN protocol semantics.
- The dual-channel IP-to-NDN translation gateway enables conducting cross-translation between TCP/IP and ICN/NDN protocol.

Another fundamental contribution of this work is the estimation of the throughput model that is comparable with emulation results in chapter 5. Through the result in

chapter 5, we realize a hit ratio degradation due to delayed hit caching in the gateway. Therefore, we explained briefly and analyzed the behavior in chapter 6. Moreover, we also proposed a method to suppress the hit ratio decline in the translation gateway.

## **7.2 Future work**

As future work, the dual-channel IP-to-NDN translation gateway still requires additional integration to global name services and its detailed fail-over prefix name binding mechanism. Furthermore, the design of hierarchical name prefix name binding for upper layer translation, such as IP-to-NDN application layer translation, is a necessary work that complements the discussion initiated in this dissertation. And the effectiveness of the suppressing method for hit ratio decline that occurs in the gateway, as mentioned in chapter 6, must be further investigated, especially regarding processing time due to complexity.

## References

- [1] A. Afanasyev et al., “NDNS: A DNS-Like Name Service for NDN,” 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, Canada, July-August 2017.
- [2] A. Hoonlor et al., “An Evolution of Computer Science Research,” *Communications of the ACM*, 56(10), pp. 74-83, Oct. 2013.
- [3] A. Mahanti, C. Williamson, and D. Eager, “Traffic Analysis of a Web Proxy Caching Hierarchy,” *IEEE Network*, 2000.
- [4] A. Padmanabhan, L. Wang, and L. Zhang, “Automated Tunnelling over IP Land: Run NDN Anywhere,” *Proceedings of the 5th ACM Conference on Information-Centric Networking*, Boston, USA, September 2018.
- [5] B. Nour et al., “Internet of Things Mobility Over Information-Centric/Named-Data Networking,” in *IEEE Internet Computing*, vol. 24, no. 1, pp. 14-24, 1 January-February 2020.
- [6] B. Nour, F. Li, H. Khelifi, H. Mounsla, and A. Ksentini, “Coexistence of ICN and IP Networks: An NFV as a Service Approach,” *IEEE Global Communications Conference (GlobeCom)*, Waikoloa, USA, December 2019.
- [7] B. Nour, H. Khelifi, R. Hussain, H. Mounsla, and S. Bouk, “A Collaborative Multi-Metric Interface Ranking Scheme for Named Data Networks,” in *International Wireless Communications and Mobile Computing (IWCMC)*, Limassol, Cyprus, June 2020.
- [8] B. Nour, S. Mastorakis, R. Ullah and N. Stergiou, “Information-Centric Networking in Wireless Environments: Security Risks and Challenges,” in *IEEE Wireless Communications*, April 2021.
- [9] B. R. Dawadi, D. B. Rawat, S. R. Joshi, P. Manzoni, and M. M. Keitsch, “Migration Cost Optimization for Service Provider Legacy Network Migration to Software-defined IPv6 Network,” *International Journal of Network Management*, vol. 31, issue 4, July/August 2021.
- [10] C. Fan, I. Chen, C. Cheng, J. Huang, and W. Chen, “FTP-NDN: File Transfer Protocol Based on Re-Encryption for Named Data Network Supporting Nondesignated Receivers,” in *IEEE Systems Journal*, vol. 12, no. 1, pp. 473-484, March 2018.

- [11] C. Guimaraes, J. Quevedo, R. Ferreira, D. Corujo, and R. L. Aguiar, "Content Retrieval while Moving Across IP and NDN Network Architectures," 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, June-July 2019.
- [12] C. Gundogan, C. Amsuss, T. C. Schmidt, and M. Wählisch, "IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison," 2020 IFIP Networking Conference (Networking), Paris, France, July 2020.
- [13] F. Fahrianto and N. Kamiyama, "Comparison of Migration Approaches of ICN/NDN on IP Networks," ICIC-APTİKOM 2020, Online, November 2020.
- [14] F. Fahrianto and N. Kamiyama, "The Dual-Channel IP-to-NDN Translation Gateway," 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2021.
- [15] F. Fahrianto and N. Kamiyama, "A Low-Cost IP-to-NDN Translation Gateway," 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), 2021,
- [16] F. Fahrianto and N. Kamiyama, "Migrating From IP to NDN Using Dual-Channel Translation Gateway," in IEEE Access, vol. 10, pp. 70252-70268, 2022
- [17] G. Carofiglio, L. Muscariello, J. Auge, M. Papalini, M. Sardara, and A. Compagno, "Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture," Proceedings of the 6th ACM Conference on Information-Centric Networking, Macao, China, September 2019.
- [18] G. Xylomenos, Y. Thomas, X. Vasilakos, et al., "IP Over ICN Goes Live," in European Conference on Networks and Communications, June 2018.
- [19] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web Caching Systems: Modeling, Design and Experimental Results," IEEE J. Selected Areas of Commun., vol. 20, no. 7, pp. 1305-1314, September 2002.
- [20] H. Kim and N. Ko, "A Scalable Pub/Sub System for NDN," 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, South Korea, October 2020.
- [21] H. Liu, K. Azhandeh, X. Foy, and R. Gazda, "A Comparative Study of Name Resolution and Routing Mechanisms in Information-Centric Networks, Digital Communications and Networks, vol. 5, issue 2, pp. 69-75, 2019.
- [22] H. Wu, J. Shi, Y. Wang, Y. Wang, G. Zhang, Y. Wang, B. Liu, and B. Zhang, "On Incremental Deployment of Named Data Networking in Local Area Network," ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, China, May 2017.

- [23] I. Moisenko and D. Oran, "TCP/ICN: Carrying TCP Over Content Centric and Name Data Networks," in ICN'16, Kyoto, Japan, September 2016.
- [24] J. Hong, T. You, and Y. Hong, "Name Resolution Service for CCN," International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2017.
- [25] J. Li, Y. Sheng, and H. Deng, "Two Optimization Algorithms for Name-Resolution Server Placement in Information-Centric Networking," *Applied Sciences* 10, no. 10:3588, 2020.
- [26] J. Thompson, P. Gusev, and J. Burke., "NDN-CNL: A Hierarchical Namespace API for Named Data Networking," In Proceedings of the 6th ACM Conference on Information-Centric Networking. Association for Computing Machinery, New York, NY, USA, 2019.
- [27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *IEEE INFOCOM*, 1999.
- [28] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, vol. 4, July 2014.
- [29] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon, "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems," *IEEE/ACM ToN*, Vol.17, NO.5, pp.1357-1370, October 2009.
- [30] M. N. D. Satria, F. H. Ilma, and N. R. Syambas, "Performance Comparison of Named Data Networking and IP-Based Networking in Palapa Ring Network," 2017 3rd International Conference on Wireless and Telematics (ICWT), Palembang, Indonesia, July 2017.
- [31] N. Atre, J. Sherry, W. Wang, and D. S. Berger, "Caching with Delayed Hits," In Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20), Association for Computing Machinery, New York, NY, USA, 2020.
- [32] P. Yue, R. Li and B. Pang, "The Random Content Poisoning Attack in NDN," *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, NY, USA, 2021.
- [33] S. Gao, H. Zhang and B. Zhang, "Supporting Multi-Dimensional Naming for NDN Applications," 2016 IEEE Globecom Workshops, Washington, DC, USA, December 2016.



- [34] S. Luo, S. Zhong, and K. Lei, "IP/NDN: A Multi-Level Translation and Migration Mechanism," IEEE/IFIP Network Operations and Management Symposium, Taiwan, April 2018.
- [35] T. Liang et al., "NDNizing Existing Applications: Research Issues and Experiences," ACM ICN 2018.
- [36] T. Refaei, J. Ma, S. Ha, and S. Liu, "Integrating IP and NDN through an Extensible IP-NDN Gateway," In Proceeding of ACM Information Centric Networking, Berlin Germany, September 2017.
- [37] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," CoNEXT 2009, Rome, December 2009.
- [38] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: Voice-over Content-Centric Networks," ReArch'09, Rome, Italy December 2009.
- [39] X. Tan, W. Feng, J. Lv, Y. Jin, Z. Zhao, and J. Yang, "f-NDN: An Extended Architecture of NDN Supporting Flow Transmission Mode," in IEEE Transactions on Communications, vol. 68, no. 10, pp. 6359-6373, October 2020.
- [40] X. Ma, UCLA, CA, USA. python-ndn(2019) [Online]  
Available: <https://zjkmxy.github.io/projects/python-ndn>.
- [41] Y. Cui et al., "Tunnel-Based IPv6 Transition," in IEEE Internet Computing, vol. 17, no. 2, pp. 62-68, March-April 2013.
- [42] Y. Liu, J. Geurts, J. Point, S. Lederer, B. Rainer, et al., "Dynamic Adaptive Streaming over CCN: A Caching and Overhead Analysis," In Proceedings of 2013 IEEE International Conference on Communications, pp. 3629-3633, Budapest, Hungary, June 2013.
- [43] Y. N. Rohmah, D. W. Sudiharto, and A. Herutomo, "The Performance Comparison of Forwarding Mechanism between IPv4 and Named Data Networking (NDN). Case study: A node Compromised by the Prefix Hijack," 2017 3rd International Conference on Science in Information Technology (ICSITech), October 2017.
- [44] Z. Yan, G. Geng, S. Zeadally, and Y. Park, "Distributed All-IP Mobility Management Architecture Supported by the NDN Overlay," in IEEE Access, vol. 5, pp. 243-251, 2017.
- [45] Z. Luo, T. Li, and Q. Zhang, "A Global Name Mapping System for ICN-IP Coexistence," ICN 2022, Osaka, Japan, September, 2022.
- [46] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: Audio Conference Tool Over Named Data Networking," ICN 2011, Ontario, Canada, August 2011.