Impact of Pending Interest Table Size and Policies on Network Performance in Named Data Networking

Madhurima Buragohain

Impact of Pending Interest Table Size and Policies on Network Performance in Named Data Networking

Thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

by

Madhurima Buragohain

under the supervision of

Prof. Sukumar Nandi

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, INDIA

May 20, 2022

In Memory of

Enaai and Late Dr. Aboni Chandra Goswami Sir

I dedicate this thesis to my beloved parents, whose blessings, sacrifice and love made my path of success

Declaration

I declare that

1. The work contained in this thesis is original and has been done by myself under the supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT Guwahati
Date: 20.1.2022

Madhurima Buragohain
Research Scholar
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, INDIA 781039

Intentionally Left Blank

Certificate

This is to certify that the work contained in this thesis entitled "Impact of Pending Interest Table Size and Policies on Network Performance in Named Data Networking " is a bonafide work of Madhurima Buragohain(Roll No. 156101017), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

Place: IIT Guwahati
Date:

Prof. Sukumar Nandi
Thesis Supervisor
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, INDIA 781039

Intentionally Left Blank

Acknowledgements

I am happy to take this opportunity to thank all the people without whom this roller-coaster journey would not have been possible. During this beautiful journey, I have discovered both my strengths and weaknesses.

First and foremost, I sincerely thank my esteemed Supervisor, *Prof. Sukumar Nandi*, for his insightful guidance, patience, and continuous support throughout my journey. It had been my long-held ambition to work under him. He gave me enough freedom, which helped me to explore different things. He taught me how to read and think differently. His enthusiasm, logical reasoning, and brilliance frequently amazed me.

I am very thankful to my doctoral committee members *Dr. Sanasam Ranbir Singh*, *Dr. John Jose* and *Dr. Ashok Singh Sairam* for providing valuable comments and suggestions to improve the work. I extend my gratitude to thesis reviewers, Prof. Swades De, Department of Electrical Engineering, IIT Delhi and Assoc. Prof. Biplab Sikdar, Department of ECE, National University of Singapore. I express my sincere thanks to former Head of the Department Prof. Diganta Goswami and Prof. S.V. Rao and current Head of the Department Prof. Jatindra Kumar Deka for providing a pleasant research environment and supporting my research work in many ways. I would also like to express my gratitude to former Director of IITG, Prof. Gautam Biswas and current Director of IITG, Prof. T.G. Sitharam, all the Deans and other officials of IITG, whose collective effort have made this institute a place for world-class studies and research. I am thankful to all the faculties and staffs of CSE department for their technical and official supports.

I gratefully acknowledge Prof. Sanjay Kumar Bose (Professor, Department of Electrical Engineering, IIT Guwahati) and Ankita Sen (Ph.D. student, Department of Mathematics, IIT Guwahati) for help in learning Queuing Theory.

I want to thank Prof. Nityananda Sarma (Tezpur University) and Mr. Biswajit Sarma (Jorhat Engineering College) for their guidance and encouragement during my MTech and BTech projects. Also, I am grateful to Dr. Rupam Baruah and Mr. Diganata Baishya (Jorhat Engineering College), who taught me the ABCD of computer science. Their enormous dedication and outstanding teaching style made me interested in CS.

I am grateful to my school and college teachers: Mrs. Gitwali Phukan (Primary school), Late Tulan Chutia, Mr. Jitu Rajkonwar, Mr. Mrinmoy Goswami, Miss Papori Rajkumari, Late Dr. Aboni Chandra Goswami (Jorhat Science College) for their significant contribution to my life.

I want to thank Prashant Gudipudi, Zaki Anwer, Chinmoy Kathar, Chinmoy Kachari for fruitful research discussion. Thank you, Anirudh Kondaveety and Sukant Dey, for always being there for me and encouraging me when I needed it the most. I would also appreciate all the support from NDN community all over the world.

I thank my labmates in Network Security Lab: Sikha, Kongkon Da, Manoj, Pinaki Sir, Subhrendu Bhaiya, Pranav Sir, Pradeep Sir, Debanjan, Saurav Gupta and Saurabh Kumar, for their cooperation. I am thankful to IITG friends: Ankita, Anasua, Kasturi, Pallavi, Dibya, Sisir Sir, Abhijit, Arunangshu, Subrata, Deepankar who made my stay at IITG enjoyable.

A heartfelt thanks to Science College friends: Gayatri Gogoi (JK), Gaytri Gogoi (Tips), Shantanu Chetia, JEC friends: Babita Pegu, Manas Baruah, Bitupan Borah for their support and help at various stages in academic life. Without their encouragement, it would not have been possible for me to get into IITG.

Last but not least, I am grateful to Almighty, Parents, In-laws, my dear Husband, Moon and all the Teachers for their unconditional love and support.

Date: 20.05.2022 **Madhurima Buragohain**

Abstract

Named Data Networking (NDN) is a clean-slate Internet Architecture where the central focus has been shifted from hosts to contents. Contents are made location-independent, and they have a unique identifier. NDN is primarily designed to address long-standing problems faced by the current Internet, such as scalable content distribution, security, mobility support etc. Communication in NDN takes place with the help of two types of packets: Interest (request) and Data (response). Consumers issue Interests with the desired content name, and routers forward these Interests towards potential producers. NDN packets do not carry any information related to consumers or producers. Therefore, a data structure (table) is designed to keep track of the forwarded Interests. Later, each node in NDN forwards the corresponding Data packets by looking at entries in the table. This table is named as *'Pending Interest Table (PIT)'* as it carries the information of unsatisfied (pending) Interests. PIT's unique design provides various advantages to NDN, such as anonymity, immediate loop detection, multipath forwarding, and multicast.

Network Forwarder Daemon (NFD) implements the NDN protocol. Although the current NFD does not have a finite PIT size, the overallocation of memory to Pending Interest Table (PIT) does not provide any benefit. Even if we allocate more memory to PIT, if there is insufficient bandwidth in the outgoing link, the outgoing buffer size may increase. This leads to more content retrieval delay and impacts delay-sensitive applications. So, PIT needs to have a fixed size. However, we have to face other challenges due to its fixed size. We can not deny the occurrence of bursty traffic in a network. Bursty traffic creates network congestion. Due to which PIT entries are satisfied slowly, this leads to more entries in PIT, and gradually, PIT may become full. It leads to degradation of Quality of Service (QoS) of the premium consumers (who pay more for better service). Moreover, the attackers can also exploit the presence of PIT to degrade QoS of the targeted legitimate consumers. In both scenarios (bursty traffic or attack), the network gets congested. So, the objective of this thesis is to study the impact of PIT size on network performance from three perspectives: QoS, security and congestion control. The first contribution of this thesis enhances the QoS of NDN using PIT replacement and PIT reservation policy. In the following contribution, we propose a smart collaborative attack model to target legitimate consumers, which exploits PIT's of the intermediate routers. Our final contribution offers a congestion control scheme that leverages PIT per outgoing face placement and explicit marking. To evaluate the performance, we implement the proposed schemes in ndnSIM simulator. We compare them with state-of-the works from literature, and based on the simulation results, we found that our proposed schemes outperform the existing approaches.

Intentionally Left Blank

Contents

List of Figures

Intentionally Left Blank

List of Tables

Intentionally Left Blank

List of Algorithms

Intentionally Left Blank

List of Abbreviations

AQM Active Queue Management

BF Bloom Filter

CBF Counting Bloom Filter

CS Content Store

CTMC Continuous Time Markov Chain

EWMA Exponentially Weighted Moving Average

FIA Future Internet Architecture

FIB Forwarding Information Base

ICN Information-Centric Networking

IFA Interest Flooding Attack

ISP Internet Service Provider

ISR Interest Satisfaction Rate

LC Legitimate Consumer

LP Legitimate Producer

MC Malicious Consumer

MN Monitoring Node

MP Malicious Producer

NACK Negative Acknowledgement

NDN Named Data Networking

NFD Network Forwarder Daemon

NPHT Name Prefix Hash Table

PIT Pending Interest Table

QoE Quality of Experience

QoS Quality of Service

RED Random Early Detection

RTO Retransmission Time Out

RTT Round Trip Time

SBP Satisfaction based Pushback

SLA Service Level Agreement

Chapter 1

Introduction

The current Internet was designed and developed during the late 1960s to communicate between computers to share hardware resources. We now primarily use it for content distribution and retrieval. According to the Cisco Visual Networking Index report [1], IP video traffic will account for 82% of total IP traffic by 2022, up from 75% in 2017. Thus, we are witnessing a significant shift in Internet usage. Furthermore, with the proliferation of content-based applications like video-streaming, e-commerce, and e-Health, the current Internet is gradually showing its limitations in security, mobility, and content distribution inefficiencies. To address these problems, Researchers have proposed overlay solutions such as Transport Layer Security (TLS), Mobile IP, Peer-2-Peer (P2P) network, and Content Delivery Network (CDN). However, due to inefficient resource utilization and layers of indirections, these solutions may not be able to fulfil the needs of future applications. With an aim to provide permanent solutions to these problems, Researchers have started looking for the root cause. They later discovered that these problems are the outcome of a mismatch between Internet usage and fundamental architecture. Fixing the mismatch between the usage and the architecture appears to be the only permanent solution to these problems faced by the Internet. In this direction, Researchers have proposed various Future Internet Architecture (FIA) proposals under the umbrella of Information-Centric Networking (ICN) [2]. Among all the proposals, Named Data Networking (NDN) emerges as one of the most popular and promising because of its simple design and widespread support from the NDN community.

The core idea of NDN is addressing contents instead of hosts. Van Jacobson, a leading TCP-IP architect, envisioned this idea in 2009 [3]. Contents in NDN are location-independent.

A name uniquely identifies them. Therefore, any node in the network can store the contents. This *'in-network caching'* feature helps to reduce content retrieval delay, producer load, and bandwidth consumption. For retrieving the desired content, a consumer sends a request (Interest packet) carrying the name of the content. As NDN packets do not carry any source or destination address, forwarded Interests are kept in a table named Pending Interest Table (PIT). The traces in PIT helps to deliver the corresponding Data packet to the requesting consumer.

PIT's unique design facilitates NDN to achieve immediate loop detection, multicast, multi-path delivery, stateful forwarding plane and anonymity. The size of PIT should be limited [4] [5]. Its overallocation does not help to improve the network performance; instead, it degrades. It happens due to an increase in the outgoing queue size, which later increases the content retrieval delay. However, due to its fixed size, in the presence of bursty traffic, it may become full. Therefore subsequent packets may be dropped, and as a result, Interest Satisfaction Rate (ISR) of the consumers degrade. ISR of a consumer is the ratio of Data packets received to the total Interest packets forwarded. It indicates the Quality of Service (QoS) perceived by a consumer. In addition to that, attackers may exploit PIT of a router by targeting to fill up the PIT so that targeted consumers' Interest packets get dropped. In both scenarios (in the presence of bursty traffic or attack), the network gets congested. So, the objective of the thesis is to study the impact of PIT on network performance from three perspectives: QoS, security and congestion control.

1.1 Motivation for the Thesis

PIT is a fundamental building block in NDN. Its unique design facilitates various advantages:

- *Anonymity:* Unlike an IP packet, an NDN packet does not contain any information about the sender and receiver. PIT keeps track of incoming and outgoing faces of an Interest packet. So, when an NDN node receives a Data packet, it forwards the packet towards downstream nodes by following those PIT traces. Due to this feature, an attacker can't target any specific consumer.

- *Stateful forwarding plane:* PIT enables NDN to have stateful forwarding plane. Because of this stateful forwarding plane, each node can measure the performance of different paths, rapidly detect failures, avoid broken links, and utilize multiple paths to mitigate

congestion.

- *Interest aggregation and multicast:* PIT aggregates the similar Interests and forwards only one Interest while remembering the interface IDs through which Interests have been received. It saves unnecessary bandwidth consumption. After receiving a Data packet, each router checks whether its PIT contains more than one incoming interface or not. If yes, it replicates the received Data packet and forwards it to all the incoming interfaces. Thus, PIT provides inbuilt multicast.

- *Immediate loop detection and multipath forwarding:* Each Interest packet contains a unique random string called '*Nonce*'. It is recorded in the PIT entry along with the Interest Name. In a network, a router may receive one Interest packet, and after some time, it may receive the same Interest through another interface. The router checks the '*Nonce*' value along with the Interest name to detect such Interest loops. Due to this loop detection, a router can forward an Interest to multiple paths without worrying about loops.

These benefits have motivated us to explore more on PIT and study its impact on network performance.

The Named Data Networking (NDN) protocol is implemented by Network Forwarder Daemon (NFD) [6]. The current NFD does not limit the size of a PIT. With the falling cost of memory, one might wonder why one should be concerned about PIT filling up. It is feasible to allocate additional memory so that PIT does not run out. However, we believe that it is a poor idea. The reasons behind our argument are as follows:

- Processing overhead (PIT lookup and update) at PIT grows with the increased number of entries.

- Overallocation of PIT does not provide any benefit if the outgoing link does not have sufficient capacity to manage the traffic.

- Oversized PIT may increase in content retrieval delay due to an increase in outgoing queue size. This impacts today's dominant real-time applications like voice and video.

A preliminary experiment in the ndnSIM simulator [7] shows that even if we do not set any limit in PIT size, we do not see any improvement in network performance. We observe the Interest Satisfaction Rate (ISR= ratio of received Data packets/total sending Interests) and the

average response time w.r.t Interest Arrival Rate. From Figure 1.2(a), 1.2(b), 1.3(a), 1.3(b) we observe that after a particular Interest Arrival Rate, the ISR decreases and the average response time increases significantly. As a result, a predetermined PIT size is required.

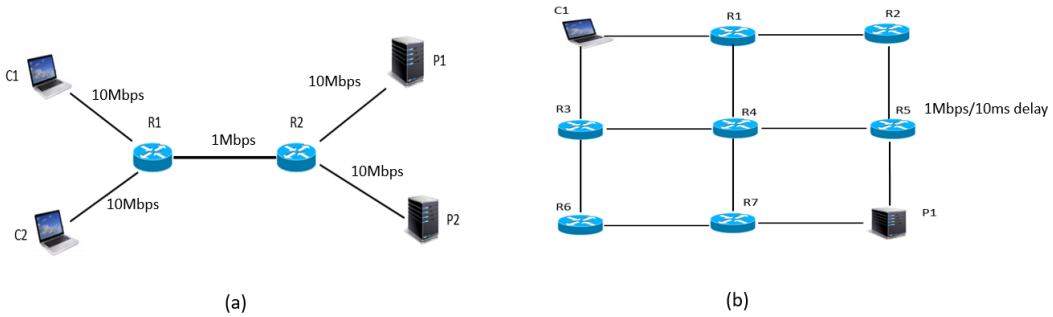
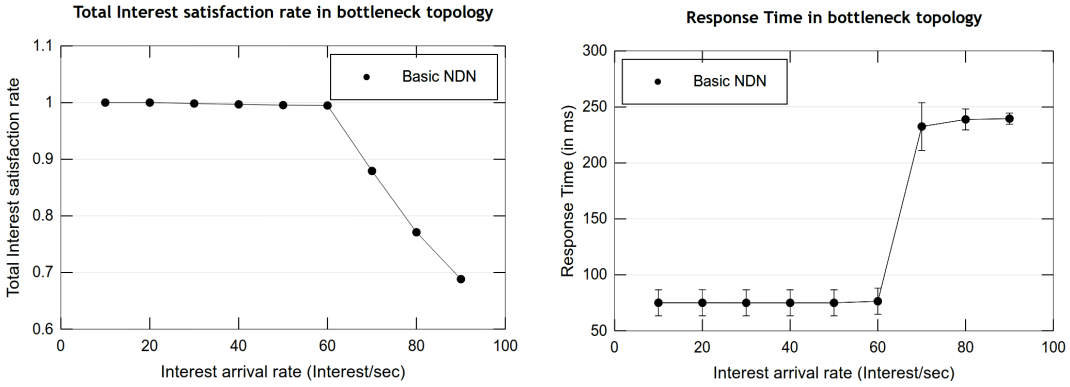
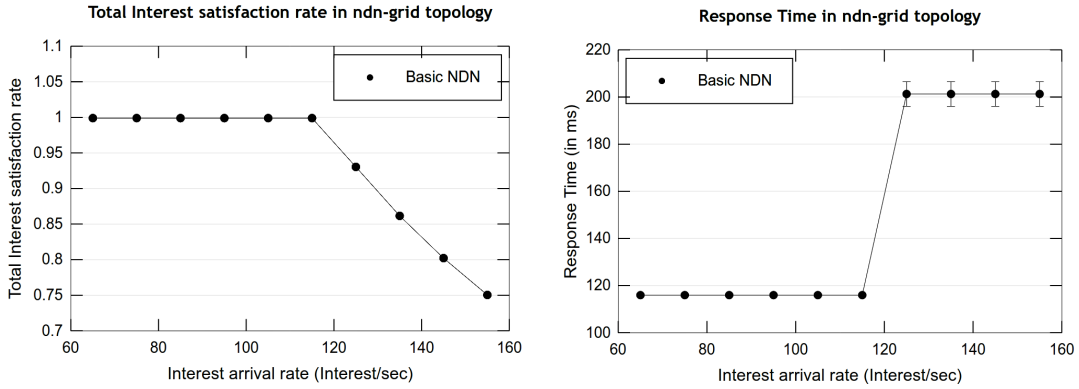


Figure 1.1: (a) bottleneck topology (b) ndn-grid topology



(a) (b)

Figure 1.2: (a) Total Interest satisfaction rate vs Interest arrival rate (b) Average Response/service time of a consumer vs Interest arrival rate in bottleneck topology



(a) (b)

Figure 1.3: (a) Total Interest satisfaction rate vs Interest arrival rate (b) Average Response/service time of a consumer vs Interest arrival rate in ndn-grid topology

However, as a result of the fixed PIT, we may face the following challenges:

- In the presence of bursty traffic, PIT may become saturated. Once it is full, subsequent packets may get dropped. It degrades the QoS of the premium customers (pays an additional cost for better service).
- Attackers can also target to degrade the QoS of legitimate consumers by exploiting PIT of the routers on the way to the producers.
- In both of the circumstances above (in the presence of bursty traffic or attackers), the network may become congested, causing the PIT to fill up.

In this thesis, we have discussed the possibility of those challenges and tried to come up with solutions to resolve them from a PIT perspective.

1.2 Contributions

This thesis discusses the impact of PIT size on network performance in NDN from three perspectives: QoS, Security, and Congestion control. The contributions of this thesis are listed in three parts, which constitute a chapter of the thesis each:

- *Contribution 1:* In our first contribution, we state the problem of QoS degradation of the premium consumers in the presence of bursty traffic. To resolve this, we propose two schemes: **PRWR** (PIT Replacement Without Reservation) and **PRR** (PIT Replacement with Reservation). We present analytical models of both the proposed schemes using a two-dimensional Continuous Time Markov Chain (CTMC) to estimate different performance metrics such as Interest packet blocking and forced termination probability at a given node in the network.

- *Contribution 2:* During the previous work, we have observed that some malicious consumers can target to fill up the PIT of NDN nodes. For real deployment, NDN should be robust and resilient to all possible attacks. Therefore, in our second contribution, we propose a smart, collaborative attack model which exploits the PIT of the routers on the way to the producers. The goal of the attackers is to degrade the QoS of the targeted legitimate consumers.

- *Contribution 3 :* In both the scenarios mentioned above (bursty traffic or attack), the network experiences congestion, which causes PIT saturation. Once the PIT is full, subsequent packets get dropped. Due to this, a consumer who sends Interests following a consumer window experiences a packet timeout. Therefore, it reduces its Interest sending rate, and thus, its goodput is reduced significantly. To address this issue, we propose a congestion control scheme that leverages PIT per outgoing face placement and explicit marking in NDN.

Our proposed methods in this thesis are evaluated in the ndnSIM simulator [7], an official simulator for NDN. To evaluate the efficacy of our proposed works, we compare them against the state-of-the-art works, and we found that they perform superior to others.

1.3 Organization

The rest of this thesis is organized as follows. Chapter 2 covers a brief introduction to NDN, which covers key features, packets, entities, naming, and packet forwarding. It also presents an inside view of Pending Interest Table (the focus of our work). Chapter 3 proposes EQPR which is designed to enhance Quality of Service (QoS) in Named Data Networking (NDN) using PIT Reservation and PIT Replacement policy. Chapter 4 proposes SCAN, a Smart Collaborative

Attack in NDN that exploits the presence of PIT in a router. Chapter 5 proposes LPECN, which is a congestion control scheme that leverages PIT placement and Explicit marking. Finally, in Chapter 6, we conclude the thesis with some possible future directions.

Intentionally Left Blank

Chapter 2

Background

This chapter briefly introduces NDN, covering its key features, naming, packets, entities, and packet forwarding. Next, we give an inside view of the Pending Interest Table (PIT) and review some PIT related works. Finally, we provide definitions for a few terminologies used throughout the thesis. It may be noted that a literature review of each contribution is included in the chapter itself.

2.1 A brief introduction to NDN

The current Internet architecture can be represented as an hourglass (see Figure 2.1), with an IP layer at the narrow waist that connects independently evolving higher application layers and the lower link layers. Any application can run over IP, and IP can run over almost all link-layer technologies. The NDN protocol stack retains the same hourglass form as TCP/IP. However, the narrow waist's functionality has been shifted from "delivering a packet to a particular host" to "retrieving Data by a name". This name can be a text message, a chunk of a book/article, a command to turn off the light bulb, or a communication endpoint [8]. This shift has been designed to address the current problems such as efficient information dissemination, security, mobility, etc. Below the narrow waist, there is a "Strategy layer". This layer optimizes the use of underlying links depending on the network conditions and other factors. Right above the waist, there is a security layer. This layer secures the content directly. Table 2.1 shows the core difference between TCP/IP and NDN architecture.

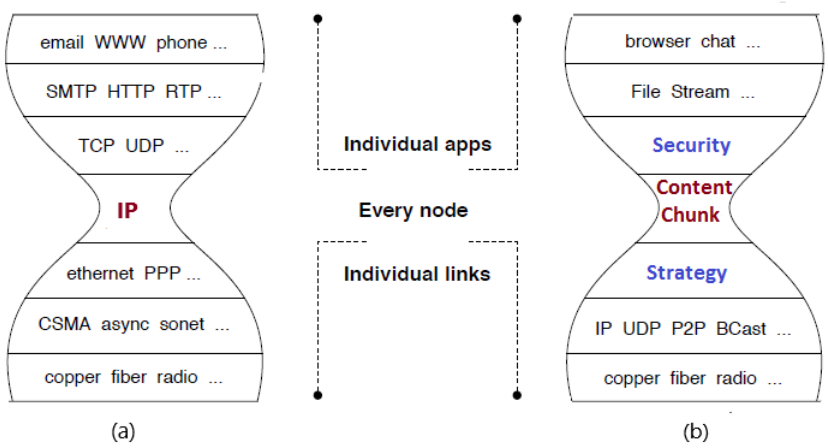


Figure 2.1: (a) TCP/IP stack and (b) NDN Stack

Table 2.1: Difference between TCP/IP and NDN architecture

Points	*TCP/IP architecture*	*NDN architecture*
Addressing	hosts	contents
Communication model	push-based	pull-based
Forwarding plane	stateless	stateful
Security	secures the communication channel	secures the content

2.1.1 Key features

The following are some of NDN's key features (refer to Figure 2.2).

- *Named and authenticated content:* Each content in NDN is identified by a unique name. Furthermore, the content is cryptographically signed by the originator. This provides *data origin authenticity* and *integrity*.
- *Receiver-driven communication model:* For content to be received, the receiver must make an explicit request.
- *In-network caching:* Contents in NDN are location-independent as their unique names directly address them. As a result, any NDN node can store contents that pass through

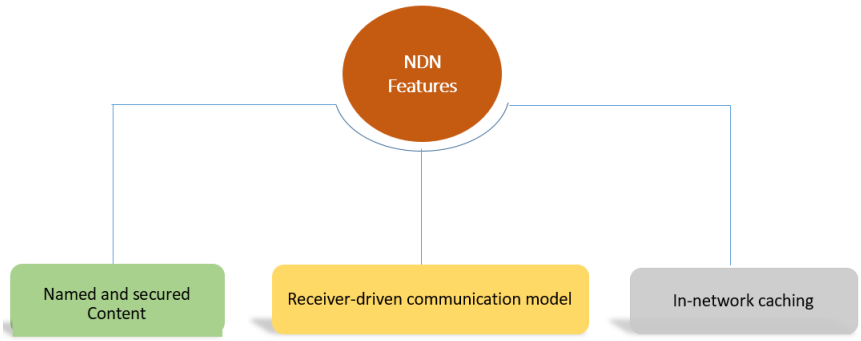


Figure 2.2: NDN Features

it to satisfy similar requests in future. Due to this in-network caching feature, there is a reduction in upstream network traffic, producer load and content retrieval delay.

2.1.2 Naming

Naming is one of the crucial components of NDN. To retrieve content, a user types related keywords, and a search application generates a list of links depending on these keywords. When the user clicks on a specific link, the application automatically sends out requests with a content name to search for the desired content. Let's say a user wishes to read an article on CNN's website. Now, the network's job is to find and deliver all of the content chunks that make up the entire article. For example, one content chunk might be named as "$/com/cnn/article.pdf/2021/Dec/ver = 1/chunk = 10 : 500$". In this case, '$/com/cnn$' is the routable prefix for the initial version of the article, and it specifies the tenth of the total 500 chunks that make up the entire article.

Along with content identification, names are used for content lookup and routing as well. Each name is hierarchical in structure and has a variable number of components separated by using the '/' character. Due to its hierarchical nature, names can be aggregated, and the longest prefix match becomes possible. The components in a name can also be of varying lengths. Applications can design their naming rule according to their requirement. For example, in an IoT application with a limited packet size, the number of components must be of lesser and shorter length. On the other hand, the applications without such constraints can use a name having more components.

2.1.3 NDN packets

There are two types of packets: Interest (request) and Data packet (response). Figure 2.3 depicts these packets.

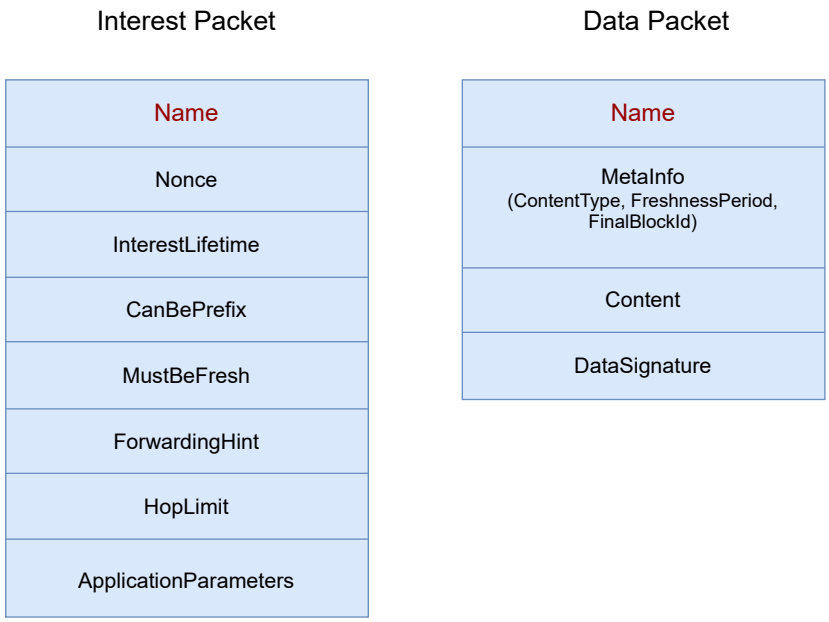


Figure 2.3: NDN Packets

Interest Packet

Different Fields of an Interest packet are described below. Name and Nonce are required fields, and others are optional.

(i) *Name:* The name of the Data chunk being requested.

(ii) *Nonce:* It is a randomly generated 32-bit string. Nonce, along with the Interest name, is also used for loop detection.

(iii) *InterestLifetime:* It shows how much time is left before the Interest expires. The application determines the value of this field based on their needs. The default value is 4 sec.

(iv) *CanBePrefix:* If this field is provided, the Name element in the Interest is a prefix, exact, or full name of the requested Data packet. Otherwise, the Name element contains the Data packet's exact or full name.

(v) *MustBeFresh:* The producer defines the FreshnessPeriod in each Data packet. When

the MustBeFresh element is absent in an Interest packet, a node can serve the satisfying Data from its cache irrespective of the validity of the FreshnessPeriod. Otherwise, a node should not return an expired Data packet.

(vi) *ForwardingHint:* A list of name delegations is stored in the ForwardingHint field. A name delegation contains a pair of names and the priority assigned to them. Each delegation means that we can access the Data packet by forwarding the Interest along the delegation path.

(vii) *HopLimit*: It specifies the maximum number of hops an Interest is allowed to be forwarded.

(viii) *ApplicationParameters:* This element can contain any arbitrary Data that is used to parameterize the Data request.

Data Packet

Different Fields of a Data packet are described below:

(i) *Name:* The name of the content in the packet.

(ii) *Meta Info:* There are three fields in Meta Info:

(a) *ContentType:* Different Content types are shown in Table 2.2.

(b) *FreshnessPeriod:* It specifies how long a node must wait after receiving Data before marking it as stale.

(c) *FinalBlockId:* It indicates the identifier of the final block in a sequence of fragments.

(iii) *Content:* This element can carry any arbitrary sequence of bytes.

(iv) *DataSignature:* It contains the signature of the content.

Table 2.2: Different Content types in Data packet

Content type	*Assigned Value*	*Description*
BOB	0	Payload identified by the Data name (default)
LINK	1	A list of delegations
KEY	2	Public key
NACK	3	Application-level NACK

2.1.4 NDN Entities

There are three types of NDN entities.

- Consumer: sends Interest packets asking for desired content.
- Producer: produces Data and then sends it in response to an Interest.
- Router: routes Interest packets and forwards corresponding Data packets.

2.1.5 Packet forwarding:

For packet forwarding process, each node relies on a system named as Network Forwarder Daemon (NFD). It has three major components: Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB). FIB is similar to the routing table in IP. However, CS and PIT are newly introduced in NDN.

- FIB: It contains name prefixes and a list of ranked interfaces for each name prefix. The major difference between IP's routing table and NDN's FIB are (i) NDN uses name prefix instead of IP prefix, and (ii) unlike IP, there are multiple entries for each name prefix.
- PIT: It stores the incoming and outgoing interfaces of each forwarded Interest. This table is necessary as NDN packets do not contain any information on source and destination. With the help of PIT, satisfying Data packets are delivered to the consumer.
- CS: It temporarily caches the received Data packets depending on the predetermined caching policy. CS helps to satisfy similar future requests and enables faster content delivery.

Figure 2.4 illustrates the forwarding process of NDN packets. Upon receiving an Interest packet, a router looks up in CS for a satisfying Data packet. If it finds, it sends the satisfying Data packet to the incoming interface through which the Interest packet has been received. Otherwise, it looks up in PIT. If there is a matching PIT entry, it means another consumer has already requested a similar Interest. In that case, the incoming Interest is aggregated by adding the incoming interface to the PIT entry's incoming interface list. If there is no match in PIT, a new PIT entry is created, and the incoming interface is added. After that router looks up at FIB using the longest prefix match algorithm. If there is a match in FIB, then Interest

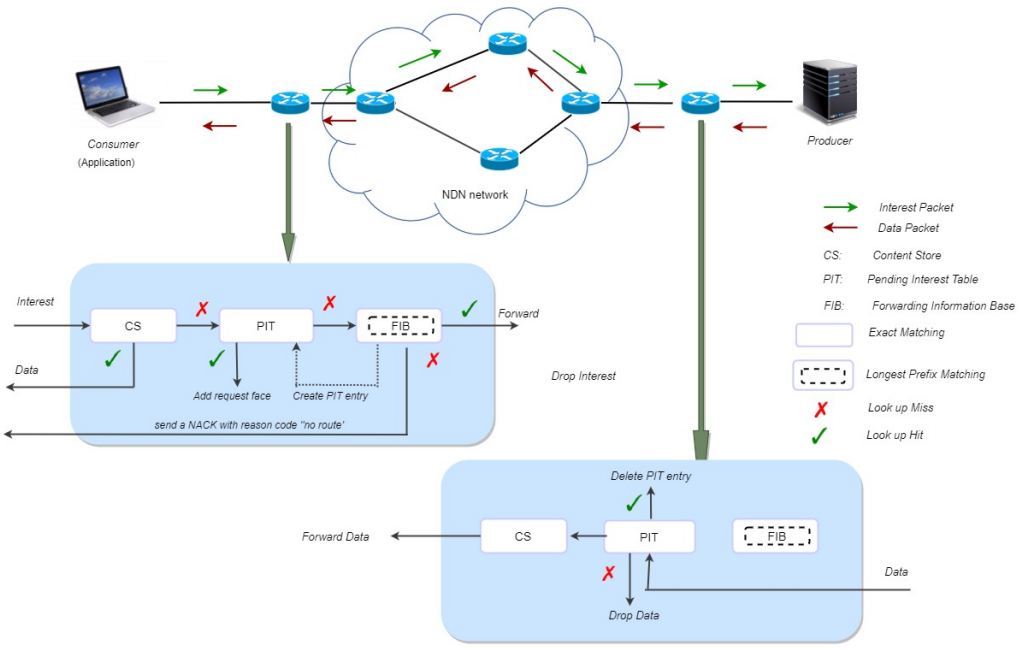


Figure 2.4: Packet forwarding process in NDN

is forwarded according to the Forwarding strategy module, and the outgoing interface is added to the PIT entry. Otherwise, the router sends a Negative Acknowledgement (NACK) towards downstream specifying the reason "No route". Each PIT entry is kept until a satisfying Data packet arrives or time out. The time out value is taken from the "Interest Lifetime field" of the Interest packet.

After receiving a Data packet, the router looks up at PIT. If there is a matching entry, it sends the Data packet to all the incoming faces listed in the PIT entry. Otherwise, it discards the Data packet considering it unsolicited due to security reasons. Afterwards, Data packets are cached in CS depending on a predetermined caching policy, and the PIT entry is deleted.

2.2 Inside view of PIT

A PIT is a collection of entries. Each entry has a list of in-records and out-records, and two timers [6]. Figure 2.5 shows the inside view of PIT

An in-record has the following information:

- a reference to the incoming face ($inFace$).

- the Nonce in the most recent Interest packet from $inFace$.
- the time-stamp of the most recent Interest packet from $inFace$.

An out-record contains the following information:

- a reference to the outgoing face ($outFace$).
- the Nonce in the last Interest packet to $outFace$.
- the time-stamp on which the last Interest packet was sent to $outFace$.
- Nacked field: this field indicates that the last outgoing Interest has been Nacked and the explanation for the Nacked status (e.g., no route, congestion, and no data).

On a PIT entry, there are two timers:

- *Unsatisfy timer:* fires when a PIT entry expires.
- *straggler timer:* fires when the PIT entry can be deleted. The deletion can be due to the reception of Data or loop detection.

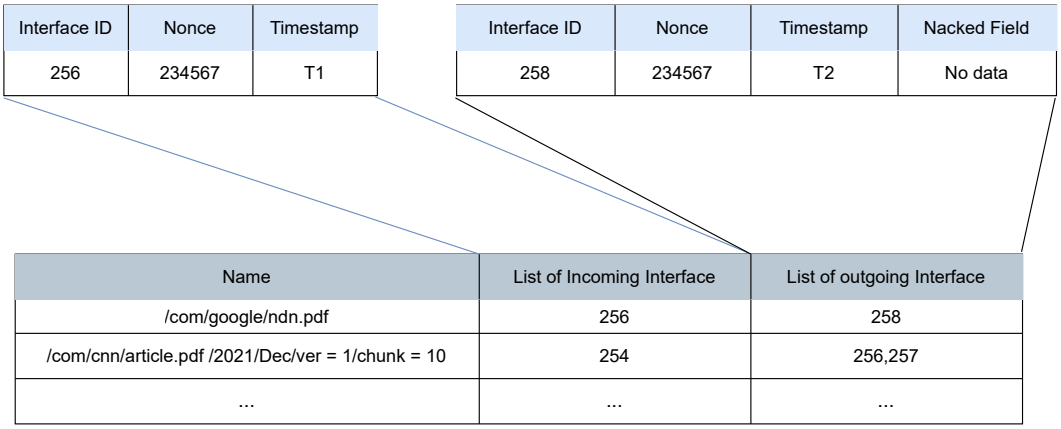


Figure 2.5: Inside view of a PIT

2.3 PIT related works

PIT is highly dynamic. When an Interest is received, either it is inserted or aggregated in the PIT. In case of Data packet arrival, the corresponding PIT entry is removed. So, for each incoming packet, the router has to perform a lookup at the PIT. For efficient implementation of PIT, we have to face the following challenges:

- *Fast Name Lookup:* PIT lookup is based on the name in the NDN packet. Though there are various existing works in the literature for IP lookup, we can not directly apply them in NDN. The reason is that the IP packet has a 32-bit address, whereas the NDN packet has variable-length and unbounded names. The longer the name, the more processing time is required.
- *Per Packet Update:* For each packet arrival, PIT needs to be looked up and updated. The PIT updation must be done within less time to match the arrival rate. Otherwise, forwarding performance is badly affected.
- *Scalability:* With the increase in line speed, the PIT needs to be scalable.

To address these challenges, the researchers have tried to propose different solutions. We have categorized the existing works into two categories: PIT Data structures and PIT placement. Subsequently, we review some of the works from the literature. Table 2.3 and Table 2.4 provides a summary of PIT data structures and PIT placement schemes.

2.3.1 PIT data structures

It refers to how PIT entries should be stored and organized to enable efficient operations. For each incoming packet, either Interest or Data, a router has to perform a lookup on PIT. With the increasing line speed, PIT has to accommodate a large number of requests. So an efficient data structure is needed so that Interest packets can be stored efficiently and operations become fast to serve the requests. Mainly three operations are performed on a PIT.

a) Insert: When a new Interest packet is received.

b) Delete: when Interest is either satisfied or timed out

c) Update: When a new incoming face is added to the existing entry.

There are three kinds of data structures to realize a PIT: Hash table, Trie, and Bloom Filter.

Yuan et al. [9] propose an approach where PIT and FIB share a hash table named Name Prefix Hash Table (NPHT). It indexes two data structures: Propagating Entries (PEs) and Forwarding Information Entries (FIEs). In both PEs and FIEs, detailed information about PIT and FIB are stored, respectively. Each bucket in NPHT has pointers to PEs and FIEs. Another data structure named *Pending Hash Table* is maintained to store all the nonce fields of the Interest packets stored in PIT. It is used for loop detection. The main drawback of the hash table implementations is an unbalanced load among its buckets. It means the length of the linked list in each bucket may differ a lot. The longer the linked list, the more time it takes to complete the lookup.

Yuan et al. [10] present an implementation of PIT using a d-left hash table. Here hash-buckets are grouped into d-subtables. When an Interest packet arrives, all the subtables are checked, and it is inserted in the subtable, which is the least loaded. Each hash table has B buckets, and each one can hold E entries. The Interests which can not fit in hash tables are inserted in an overflow table. To reduce the memory and search time, Interests are stored as fixed fingerprints instead of strings. Though the overflow table is designed to resolve the bucket overflow, it slows down the searching time. Another disadvantage of this approach is that the probability of having duplicated fingerprints (corresponding to distinct keys) mapped to the same hash bucket is not negligible. Interest packets with the same name can not be aggregated in their design. Interest aggregation is relaxed in core routers to guarantee packet delivery in case of fingerprint collision. On the other hand, edge routers support Interest aggregation and minimize most of the traffic for core routers.

DiPIT [11], maintains one PIT per incoming face. For example, a PIT corresponding to $face\_id = 256$ stores the Interest packets received from incoming face 256. Each PIT is implemented using Counting Bloom Filter (CBF). Furthermore, to lower the false-positive rate generated by each Bloom Filter (BF), all faces share a central BF. According to the authors, it uses 63% less memory and has a higher throughput than a hash table implementation.

A BF can simply remember the footprint of an Interest. It is not used for storing other information. Whereas, in the case of a PIT, along with the Interest name, the incoming face and outgoing face are also recorded. So to tackle this problem, they have provided one PIT per face. Another fact of using CBF instead of standard BF is that the entry needs to be removed when the Interest is satisfied. Since the standard BF has no provision to remove an element,

they have gone for CBF. The drawback of this scheme are : (a) In case of false positives, Data packet transmission is wasted (b) removal of expired entries through one counter per PIT does not seem valid.

MaPIT [12] uses Mapping Bloom Filter (MBF), which is a modified version of BF. MBF consists of two components: an index table and a packet store. Index table and packet store are deployed in on-chip and off-chip memory, respectively. The Packet store contains the information of the NDN packets, and the job of the index table is to access the packet store. There are two structures in the index table: A regular BF and a mapping array (MA). MA's value is used as an offset to access the packet store. In this approach, multiple distinct keys may have the same MA value (collision). There is no discussion on how to resolve them if it occurs.

In BFAST (Bloom Filter-Aided haSh Table) approach [13], a CBF is used to balance the load among hash table buckets. It applies a unified index approach to reduce repeated lookups for each table FIB, CS, and PIT. Each index entry has a pointer field that points to an entry in one of these tables. In addition to that, an associative type field is used to specifically mention the table. The drawback of this scheme is that pointers induce additional memory overhead.

Dai et al. [14] propose a scheme named Encoded name Prefix Trie (ENPT), where each component is encoded to a 32-bit integer using a code allocation function. However, this approach has several drawbacks. First, it needs a new encoding scheme. Second, it requires additional memory. Third, for mapping codes to components, it needs a hash table. Furthermore, it does not discuss loop detection and removal of expired PIT entries.

Table 2.3: Summary of Different data structures for PIT

Research Paper	Approach	Pros	Cons
NPHT [9]	Name prefix hash table	simple	Unbalanced load among buckets.
Yuan et al. [10]	d-left hash table	Less memory	Slows down the searching time
DiPIT [11]	Counting Bloom Filter	Less memory Consumption and Fast lookup	Can not remove entries when time out
MaPIT [12]	Mapping BF	Efficient in memory and Time	Loop Detection is not possible Entry Lookup has to check on every subtable Does not discuss packet processing time and collisions in packet store.
BFAST [13]	Counting Bloom Filter	High lookup, throughput and low latency	Does not discuss loop detection
ENPT [14]	Encoded name prefix tree	Performs well in case of popular contents	Additional Encoding Algorithm is required Consume more memory space Does not specify any mechanism to detect loop Removal of expired entries is not discussed

2.3.2 PIT placement

A major issue in the implementation of a PIT is its placement in the NDN router. In basic NDN, it is assumed that PIT is a single entity. In case of high traffic, especially in core routers, PIT has to accommodate millions of entries. Moreover, the operations on PIT needs to be very fast. So, it needs to be deployed in a single on-chip memory. A large number of entries in PIT may not fit in a single chip, and deploying in multiple chips would require coordination, which may affect performance. Researchers have proposed to segregate the PIT and place it at each interface of the router. PIT placements are categorized into four types [15]: (i) Input-only (ii) Output-only (iii) Input-Output and (iv)Third-party

(i) Input-only placement: In this placement, PIT is placed on each input line card [11]. An incoming Interest is entried in the PIT of the incoming interface. There are a few flaws with this placement. First, after receiving a Data packet, to forward it towards downstream, a router has to lookup the PIT. Therefore, in this placement, the router broadcasts the Data packet to all other input line cards. Second, each PIT is only aware of the Interests received by its respective interface, so Interest aggregation is not possible. Third, a router may receive an Interest from one interface, and after some time, it can loop back through another interface. Therefore, a loop detection problem exists.

(ii) Output-only placement: Di et al. [14] propose the idea of placing PIT in the outgoing line card. After receiving an Interest packet, a router does not create any PIT entry at the input line card. Only the entry is recorded on the output line card. The output line card is decided by using the longest prefix match algorithm in FIB. Though this approach satisfies interest aggregation, it has several limitations. First, if an Interest is forwarded through multiple interfaces, it may forward duplicate Data packets since it is impossible to know whether the same Data packet is already forwarded. Second, it also suffers from a loop detection problem. Third, for similar Interests, each one needs FIB lookup.

(iii) Input-Output placement: Here, PIT is placed in both input and output line cards. For each incoming Interest, there are two PIT entries: one at the input line card where it is received and the other at the output line card where it is forwarded. This placement does not satisfy Interest aggregation, multipath and loop detection. In addition to that, for each received Data packet, a router has to perform two lookup operations: (i) in the PIT of the input line-card and (ii) the PIT(s) of the output line-card(s).

(iv) Third-Party placement: Unlike discussed above, in this placement, upon arrival of an Interest, PIT entry is not created corresponding to the incoming or outgoing interface. PIT entry is created in a delegated interface [15]. This interface is determined using a hash function with the name as input. The output of the function determines the delegated interface where the entry is made. In the case of Data packet arrival, using the same hash function, the delegated interface can be determined. Since the delegated interface is decided directly from a name, only PIT lookup is required for each incoming Interest and Data packet. Due to the same reason, this placement satisfies Interest aggregation and loop detection. However, this placement requires additional *switch fabric* to switch from the incoming interface to the delegated interface. It causes hardware overhead.

Table 2.4: Drawbacks of different types of PIT Placement

PIT placement	Drawbacks
Input-only	a) After a Data packet arrival, we have to search all PITs corresponding to different incoming interfaces.
	b) Does not support Interest aggregation.
	c) Loop detection problem.
Output-only	a) For similar Interests from different faces, FIB needs to be accessed multiple times.
	b) No provision for loop detection
	c) Multiple receptions of Data in case of multipath forwarding.
	d) Does not fully satisfy interest aggregation.
Input-Output	a) Loop detection problem
	b) Multi-path issue
	c) Requires two PIT lookup per Data packet.
Third-party	Additional Switch fabric needed causing hardware overhead

2.4 Definition of some terms related to NDN

- *Upstream (forwarding):* Forwarding packets in the direction of Interests (i.e., Interests are forwarded upstream): consumer → router → router → ... → producer [16].
- *Downstream (forwarding):* Forwarding packets in the reverse direction of Interest forwarding (i.e., Data and Interest Nacks are forwarded downstream): producer → router → ... → consumer(s) [16].
- *Interest aggregation:* In this process, multiple Interest packets having the same name and

other parameters are combined together. So, due to aggregation, there is only one PIT entry for multiple requests for the same content. [16].

- *Interest Satisfaction Rate:* It is the ratio of the total number of received Data packets to the total number of forwarded Interest packets.

- *Stateful forwarding plane:* Each router stores the state of each forwarded Interest packet so that later it can deliver received Data packets back to the consumer. The recorded information is also used for measuring the performance of outgoing links, which helps to adjust the packet forwarding decision. This is different from the Current Internet, where once the packet is forwarded, routers immediately forget the information [17].

- *Symmetric forwarding:* A Data packet is forwarded through the reverse path of the corresponding Interest packet [18].

Chapter 3

EQPR: Enhancing QoS in NDN using PIT reservation and PIT replacement policy

PIT is a core building block in NDN, which facilitates various advantages such as anonymity, multicast and multipath forwarding. We discuss in section 3.1, the importance of PIT size limitation. In the presence of bursty traffic, the network may experience congestion, and as a result, PIT may be full. In that case, a router has two options: (i) simply drop the incoming Interest packet or (ii) replace an existing PIT entry to make room for the incoming one. Each option has its own drawbacks. In option (i), premium consumers (who pay extra cost for better service) also get dropped. It results in lower Interest Satisfaction Rate (ISR) of the premium consumers. In the case of option (ii), the independent decision of PIT entry replacement results in more degradation of ISR of the consumers. It is because eviction of PIT entries erases the state information of forwarded Interest packets. As a result, later even after the reception of Data packets, we can not forward the Data packets downstream. It causes degradation of ISR and finally impacts QoS. To address this issue, in this chapter, we propose two simple approaches, PRWR and PRR, with an aim to enhance QoS.

The contributions of this chapter are as follows:

- We propose two schemes to provide QoS to the premium users using PIT replacement and PIT entry reservation policy.
- With an aim to reserve PIT entries for premium consumers' Interests, we propose a simple and dynamic PIT entry reservation method.
- We present analytical models of both the proposed schemes using a two-dimensional Continuous Time Markov Chain (CTMC). We estimate performance metrics such as Interest packet blocking and forced termination probability at a given node in the topology.
- We use simulation results to validate the analytical model. We study the relation between Interest arrival rate and performance metrics.

The remainder of this chapter is organized as follows: Section 3.1 covers motivation behind our proposed work, section 3.2 presents related works, section 3.3 presents network model. The detailed description of the proposed schemes are presented in section 3.4. Section 3.5 covers analytical modeling of the proposed schemes. Section 3.6 presents performance analysis and finally summary is discussed in section 3.7.

3.1 Motivation

Currently, in NFD, there is no size limit in PIT. With the decreasing cost of memory, a simple question may come to our mind: Should we worry about PIT filling up? Can't we ensure PIT never runs out of memory? We have three reasons to argue in favour of limiting PIT size. First, as PIT entries grow, the PIT processing overhead increases. Second, even if we allocate more memory, the outgoing link may not have enough capacity to carry the traffic. It helps to grow the size of PIT drastically. Third, oversized PIT aids in increasing the queuing buffer's occupancy, which negatively impacts the performance of delay-sensitive applications. To support our argument, we have conducted a preliminary experiment to study the relation of ISR and Response time w.r.t. increase in Interest arrival rate. The results show that after a certain Interest arrival rate, there is a sharp decline in ISR. However, we observe a sudden increase in response time at the same point (We have already discussed in section 1.1, refer to Figure 1.2(a), 1.2(b), 1.3(a), 1.3(b)). So, we realise that having infinite size does not benefit us. As a

result, we need a fixed PIT size. However, if we fix the PIT size, we face other consequences. We can not deny the occurrence of bursty traffic in a network. Sometimes, it may cause the PIT to fill up. In that case, packet dropping takes place, which results in a reduction in consumer's ISR and degrades QoS of the consumers. Our goal is to enhance QoS using PIT reservation and PIT replacement scheme.

3.2 Related Works

In this section, we briefly review PIT replacement policies and Round Trip Time (RTT) estimation mechanisms as they are primarily related to our proposed work.

3.2.1 PIT replacement policy

RAPIT [4] presents a PIT replacement policy that is designed to find and delete non-responded PIT entries. To attain this goal, they have defined the parameter named *'importance'* for incoming Interest and PIT entry. When a router receives an Interest, it compares the importance of the Interest and the PIT entry having the lowest importance. Router evicts the one which has the lowest importance. To determine *'importance'* of a PIT entry (I_{pe}), they have defined *Timeout judgement coefficient (η)*, which describes the likelihood of a PIT entry being non-responded (i.e., timed out). η = remaining lifetime / Pending Entry Lifetime). The value of η is used to estimate I_{pe}. Suppose η is less than the timeout judgment threshold δ. In that case, the router can deduce that there is less probability of receiving content back as it is taking substantially longer than typical RTT. In RAPIT, they have considered δ =1/3. If $\eta < \delta$, the importance of the PIT entry (I_{pe}) is set to η. Otherwise, it is set to 1. For an incoming Interest, *'importance'* is calculated depending on network condition and the type of Interest. If the Interest is used for RTT measurement, the importance of that Interest is set at a high value. If the Interest is normal, in that case, further the network condition is checked. If the network condition is good, then importance is set as 1; otherwise, it is set as δ.

Hassan et al. [19] investigate the popular cache replacement policies such as Least Recently Used (LRU), First in first out (FIFO), and Least Frequently Used (LFU) in PIT. However, applying the policies in PIT result in a significant decrease in ISR of the consumers due to frequent PIT entry replacements.

HLLR [20] proposes a replacement policy that replaces the entry with the smallest number

of incoming interfaces and the maximum lifetime. The biggest drawback of this method is that it may evict an entry with only a few milliseconds to receive the Data packet.

3.2.2 RTT estimation mechanism in NDN

Due to the 'in-network caching' feature in NDN, Data packets may come from any intermediate nodes. No one can give a guarantee from which node the next Data packet may arrive. Therefore, we can not apply TCP/IP RTT estimation mechanism in NDN directly. A few RTT estimation schemes are available in the literature.

Tan et al. [4] propose a RTT estimation approach where each node measures RTT for each producer after each time interval (T_{msr}). The value is stored in FIB. When a node receives an Interest, it checks whether the RTT measurement stored in the FIB is fresh or not. If the measurement has already expired, it includes its ID and sets an RTT measuring flag to the incoming Interest and forwards it to the interfaces by looking up at FIB. Any intermediate routers do not serve these Interests; only producers serve them. Producer includes RTT measuring flag and node ID (that are carried in the Interest packet). After receiving a Data packet, a router checks the RTT measuring flag. If it is 1, it matches the node ID contained in the Data packet with its ID. If the result is positive, then it calculates the RTT. Later it removes the flag and forwards it downstream.

CCTCP [21] maintains multiple RTT values for each producer. It sends anticipated Interests along with the original Interests to know the producers beforehand. It adds additional overhead to the consumers.

ICP [22] calculates the response delay for each received Data packet. The response delay is the time interval between issuing an interest packet and receiving a Data packet. RTO is calculated as: $RTO = RTT_{min} + (RTT_{max} - RTT_{min}) * \delta$. for each flow ($\delta$=0.5). RTT_{min} and RTT_{max} are averaged across a sample history that excludes retransmitted packets. After measuring the first round trip delay, we set RTO=10 ms and $RTT_{max} = 2RTT_{min}$.

3.3 Network Model

We consider that there is a Service Level Agreement (SLA) between Internet Service Provider (ISP) and its consumers. To obtain better service, some consumers may wish to pay extra costs. The requests of these premium consumers are termed prioritized Interests, while others

are termed non-prioritized Interests. Consumer-side edge router differentiates between these Interests by marking the prioritized Interests. We add one field named 'priority' in the Interest packet to simplify the marking process. It is possible because of the Type-Length-Value encoding of Interest packets. We set the default value of the priority field as 0. It is set to 1 once the edge router decides to mark the Interest packet. Other routers on the path between consumer and producer take action depending on the value of the priority field. So, in simple terms, we consider an Interest packet to be prioritized if its priority field is set to 1; otherwise, we consider it non-prioritized.

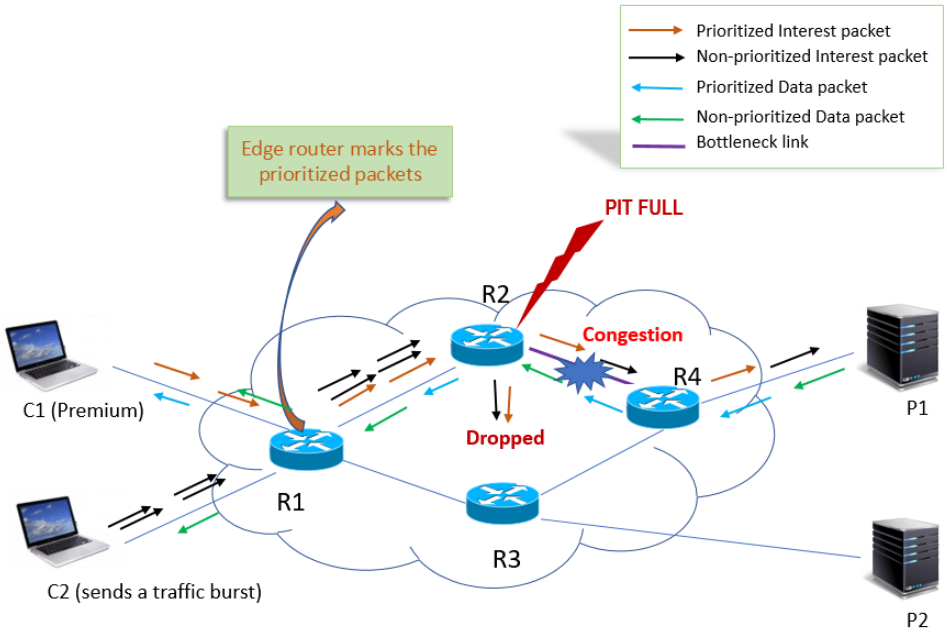


Figure 3.1: System model of EQPR

Figure 3.1 illustrates the system model where there are two consumers $C1$ and $C2$. $C1$ is a premium consumer, and $C2$ is a normal consumer, which sends a traffic burst. R3-R4 is a bottleneck link. $C1$ sends Interests to producer $P1$ which traverses path from $C1 \rightarrow R1 \rightarrow R2 \rightarrow R4 \rightarrow P1$. $C2$ also sends Interests that traverse through $C2 \rightarrow R1 \rightarrow R2 \rightarrow R4 \rightarrow P1$. Due to traffic burst sent by $C2$ and bottleneck ink R3-R4, the queuing delay at router R2 increases. It results in an increase in content retrieval delay. So, entries can stay more time in PIT and gradually, PIT become full. Therefore, further incoming Interests get dropped due to

the unavailability of space in PIT. This lowers the ISR of premium consumer C1 and affects QoS. To address this problem, we proposed two schemes that use PIT reservation and PIT replacement policy. We discuss the proposed schemes in the next section.

3.4 Proposed Schemes

We propose two schemes: PRWR (PIT Replacement Without Reservation) and PRR (PIT Replacement with Reservation) to enhance QoS in NDN. The details are discussed next.

3.4.1 PRWR scheme

Algorithm 1 depicts the formal description of the PRWR scheme. Each router keeps track of its current PIT occupancy (c). So, when a router receives an incoming Interest, it checks whether c is less than the total PIT size (N). If the result is yes, then it allows entry into PIT. On the other hand, if the PIT is already full, then it further checks if the incoming Interest is prioritized or not. For that, the router checks the *'priority'* field of the Interest packet (refer to *getPriority()* in Algorithm 1). If it is prioritized, then it finds a victim entry. As we can see in Algorithm 1, *FindEntryToBeReplaced()* tries to find the victim entry. In case the algorithm successfully finds the victim entry, the prioritized Interest is allowed to make an entry into PIT by replacing the victim entry. The process of victim entry selection is discussed in subsection 3.4.3.

Even though the PRWR scheme looks simple and successful in providing QoS to premium consumers, in practice, the frequent PIT entry removal leads to lower ISR. The reason is that a router can not forward received Data packets downstream if their state at PIT is already removed. To address this problem, we propose a PRR scheme, a variant of the PRWR scheme.

3.4.2 PRR scheme

The fundamental idea of this scheme is to reserve a dynamically adjustable minimum number of PIT entries (K) for prioritized interests. The reservation lowers the number of replaced PIT entries. However, it increases the number of blocked non-prioritized interests. Now, the question comes: what should be the value of K? The value of balances between the number of replaced entries and the number of non-prioritized blocked Interests. The main objective is to increase the overall satisfaction rate. Algorithm 2 illustrates the PRR scheme.

Algorithm 1: PRWR scheme

Input: *interest* /* incoming Interest*/, N /* PIT size (in terms of number of entries)*/

Output: Entry into PIT/drop

```
1  function Forwarder.onIncomingInterest (interest)
2  c= getCurrentPITsize()  // getCurrentPITsize() function retrieves the current value of
     PIT size
3  if c < N then
4      Allow interest to entry into PIT
5  else
                          // getPriority() function retrieves the priority of Interest packet
6      if interest.getPriority()==1 then
7          victim =FindEntryToBeReplaced (interest)          // calls Algorithm 4
8          if victim!= NULL then
9              Allow interest to entry into PIT by replacing victim
10         else
11             Drop Interest
12     else
13         Drop Interest and send a NACK with reason code 'PIT FULL'
```

When a router receives an Interest, it checks whether current PIT occupancy (c) is less than $N - K$ or not. If it is yes, the Interest can get in PIT. Otherwise, it checks the condition: $N - K \leq c < N$. If it satisfies the condition, it further checks the priority of the Interest. If the Interest is prioritized, it can get an entry into PIT. Otherwise, it is dropped, and a NACK is sent downstream to notify the consumer. If the condition: $N - K \leq c < N$ becomes false, a victim entry is selected for replacement. The selection of victim entry is explained later in subsection 3.4.3.

The formal description of the estimation of K is illustrated in Algorithm 3. Though we can use a static value of K, it is better to estimate it in real-time according to requirements due to the dynamic nature of the traffic. As shown in Algorithm 3, we consider that we have the

Algorithm 2: PRR scheme

Input: *interest* /* incoming Interest*/, N /* PIT size (in terms of number of entries)*/, K /* number of reserved PIT entries for prioritized Interest */

Output: entry into PIT/drop

```
1  function Forwarder.onIncomingInterest (interest)
2  c= getCurrentPITsize()  // getCurrentPITsize() function retrieves the current value of
     PIT size
3  if c < N − K then
4      Allow interest to entry into PIT
5  else if (N − K) ≤ c < N then
6      if interest.getPriority()==1 then
7          Allow interest to entry into PIT
8      else
9          Drop Interest and send a NACK with reason code 'PIT FULL'
10 else
11     victim =FindEntryToBeReplaced (interest)          // calls Algorithm 4
12     if victim!= NULL then
13         Allow interest to entry into PIT by replacing victim
14     else
15         Drop Interest
```

information of Maximum tolerable non-prioritized forced termination probability (Q_{max}). Q_{max} is the probability with which a non-prioritized PIT entry is replaced when a prioritized Interest arrives and PIT is already full. The value of Q_{max} can be calculated theoretically, which we discuss in section 3.5.

For each time interval of τ seconds, each node counts the number of replaced PIT entries (R) and the total Incoming Interest packets (I). Then it estimates the ratio R/I and compares it to Q_{max}. If it finds R/I is greater than Q_{max}, then it sets K to 1. In the next step, the router uses Exponentially Weighted Moving Average (EWMA) method to estimate R_n and I_n as shown in

Algorithm 3: Algorithm for estimation of reserved number of PIT entries for prioritized Interests

Input: Q_{max} /* Maximum tolerable non-prioritized forced termination probability */

Output: K /*Minimum number of PIT entries to be reserved within current Interval*/

Step 1: Calculate the number of replaced PIT entries R and the total number of Interest packets I for each time interval τ seconds.

If $R/I \geq Q_{max}$, then $K = 1$;

Step 2: Once $R/I \geq Q_{max}$,

Estimate the anticipated value of R and I for the next τ seconds using exponentially-weighted moving average (EWMA)

$R_n = (1 - \alpha) * R + \alpha * R_{n-1}$

$I_n = (1 - \alpha) * I + \alpha * I_{n-1}$

where $R_n = n$th estimate of replaced PIT entries, $I_n = n$th estimate of arrived Interest packets, R = number of replaced PIT entries in current time interval, I = number of Interests arrived in current time interval and α = Balancing factor ($0 \leq \alpha \leq 1$)

Step 3: Calculate $value = R_n/I_n$

Step 4: If $value = 0$, check the value of K.

If $K > 1$,then $K- = 1$;

Step 5: If $value \geq Q_{max}$ otherwise $K+ = 1$;

Step 2 of Algorithm 3. Next we evaluate the $value = R_n/I_n$. If we found $value$=0, we further check the value of K. If it is greater than 1, we decrement the value of K by 1. Otherwise, we compare it with Q_{max}. If it is more than Q_{max}, then we increment the value of K by 1.

3.4.3 PIT entry replacement policy

Though a rich literature is already available on cache replacement policy [23], a key question arises: why do we need a new replacement policy? Can we reuse a cache replacement policy? The answer to this question is '*No*'. Each node's independent decision of PIT entry removal further degrades the consumer's ISR. The reason behind it is that Data packet forwarding depends on the information stored on the PIT. When we remove an entry, even though a Data packet is received in a router, it can not forward the Data packet. Therefore, it discards the

Data packet. With the increase in hop length, there can be more number of dropped Data packets which result in lesser ISR.

Algorithm 4 demonstrates the victim entry selection. The selection is based on two parameters: *Pending Entry LifeTime (PEL)* and *expected RTT*. We select the entry such that meets the following criteria: *Pending Entry LifeTime (PEL)* $<$ *expected RTT*. This PEL value is chosen by applications based on their own requirements. Application set this value in *'InterestLifeTime'* field of an Interest packet. The default value is 4 sec. The estimation of *expected RTT* is discussed in subsection 3.4.4.

Algorithm 4: *FindEntryToBeReplaced* (*interest*)

1 ***Step 1:*** return a non-prioritized PIT entry which satisfies the criteria: *Pending Entry LifeTime (PEL)* $<$ *expected RTT*

2 ***Step 2:*** If step 1 fails, return a non-prioritized PIT entry having a minimum value of R (R=$sending\_time + PEL - current\_time$)

3 ***Step 3:*** If step 2 fails, return *NULL*.

If no PIT entry meets the above criteria (*Pending Entry LifeTime (PEL)* $<$ *expected RTT*), then a non-prioritized entry with a minimum value of R is selected. The calculation of R is as follows: R=$sending\_time + PEL - current\_time$. R defines the waiting time for data packets.

Now, another question may come to our mind: why do we consider the minimum, not the maximum? It is because PIT entry has been given sufficient time to fetch a Data packet. An application sets the PEL value by two or three times the RTT in most cases. As a result, there is a lower chance of satisfying a PIT entry with a minimum R value.

3.4.4 Our proposed RTT estimation approach

In NDN, estimation of RTT becomes complicated due to the presence of 'in-network caching'. Though a few proposals are already available in state-of-art literature (refer to section 3.2.2), the accuracy comes with complexity. So, we propose an RTT estimation approach in which each router measures RTT for each name prefix between itself and the nearest content producer. For smooth calculation, we have added some new fields to the Interest and Data packets. Moreover, we have added additional fields in FIB for RTT estimation. The fields along with the description are mentioned in Tables 3.1 and 3.2. The formal description of our proposed RTT estimation

scheme is described in Algorithm 5.

When a router receives an Interest packet, it first examines the corresponding FIB entry's $rttExpireFlag$ (refer to line number 3 of Algorithm 5). This field indicates that the RTT for that name prefix has not been calculated, or the calculated RTT has already expired (the RTT expiration timer is set earlier). If the value of $rttExpireFlag$ is 1, before forwarding the Interest packet to the outgoing interface, the router sets $RttMeasuringIFlag$ and $RttFlag$ to 1 (refer to line numbers 4 and 5 of Algorithm 5). By setting $RttMeasuringIFlag$ equal to 1, the router informs other nodes that this Interest packet is special and is being used for RTT measurement. If $RttFlag$ =1, it signifies that RTT measuring Interest is already forwarded. For calculation of RTT, the router records the timestamp at which Interest is forwarded (refer to line number 6 of Algorithm 5). If the $rttExpireFlag$ is 0, it checks its Content Store (CS) for satisfying Data packet. If it is yes, then it sends the Data packet, including the estimated RTT for that prefix. Otherwise, the Interest is forwarded to the next hop.

Table 3.1: New tag fields added in Interest and Data packet

Field	Description
$RttMeasuringIFlag$	If the value of this field is 1, then the Interest packet is being used to calculate RTT. Otherwise, it is a normal Interest
$RTTmeasuringDFlag$	The Data packet having this field (set to 1) is used for RTT measurement.
$RttValue$	Other routers use this field to calculate their own RTT.

After receiving a Data packet, the router first checks $RTTmeasuringDFlag$ (refer to line number 15 of Algorithm 5). If it is 1, a Data packet is used for RTT measurement. After that, the router verifies the $RttValue$ field. If it is not NULL, that means we can use RTT calculation from other intermediate users (refer to line number 19 of Algorithm 5). Otherwise, we can simply calculate as: $time :: now()$- $FibEntry.sendTime$ (refer to line number 21 of Algorithm 5).

Our proposed RTT estimation scheme has some similarities with RAPIT [4]. However, the significant difference between them is as follows: (i) When a router requires an RTT estimate in EQPR, the RTT measuring Interest is not always sent to the producer; instead, if the node has an unexpired RTT, it is satisfied by the in-network caches, and the estimated RTT is provided

Algorithm 5: RTT estimation in NDN

Input: *interest* /* incoming Interest*/, N /* PIT size (in terms of number of entries)*/, K /* number of reserved PIT entries for prioritized Interest */

Output: entry into PIT/drop

```
1  function Forwarder.onIncomingInterest (interest)
2  FibEntry ← FIB.Find (interest.getPrefix())
                                  // getPrefix() function returns the name prefix of the Interest
3  if FibEntry.rttExpireFlag == 1 then
4      interest.RttMeasuringIFlag=1
5      FibEntry.RttFlag=1
6      FibEntry.sendTime=time::now()
7      forward the Interest
8  else
9      if ContentStorehit then
10         data.RttValue=FibEntry.mrtt
11         send the Data packet
12     else
13         forward the Interest
14 function OnReceiveData (data)
15 if  data.RttMeasuringDFlag==1 then
16     FibEntry ← FIB.Find (Data.getPrefix()) if FibEntry.RttFlag == 1 then
17         if data.RttValue! = NULL then
18             FibEntry.mrtt =data.RttValue + time::now()- FibEntry.sendTime
19         else
20             FibEntry.mrtt ← time::now()- FibEntry.sendTime
21
```

Table 3.2: New fields added in FIB)

Field	Description
RttFlag	If it is set that means RTT measuring Interest is forwarded.
mrtt	This field stores the value of the calculated RTT for each name prefix
rttExpireFlag	This field indicates that the RTT value has expired or that the calculation has not yet been completed. It expresses the requirement for a new calculation.
sendTime	Contains the sending time of the RTT measuring Interest packet.

together with the Data packet. In RAPIT, however, RTT measuring interest is always served by the producer.

(ii) EQPR calculates RTT for all nodes, which uses a single RTT measuring interest on the path to the producer (in case the nodes require an RTT estimate). RAPIT uses a single RTT measuring Interest to compute RTT for only one node.

3.5 Analytical modeling of PRWR and PRR scheme

We consider an NDN node with a PIT size of N entries. The node receives Interest packets at a rate λ per second following a *Poison process.*

Our assumption: $\lambda=\lambda_p + \lambda_{np}$

where λ_p and λ_{np} are the prioritized and non-prioritized Interest packet arrival rate respectively.

The service time duration of prioritized and non-prioritized interests follow a *negative exponential distribution* with mean duration $1/\mu_p$ and $1/\mu_{np}$ respectively.

We consider all re-transmitted packets as new packets.

Table 3.3 contains the notations that are used throughout the chapter.

The rate at which PIT entries are made is calculated as follows:

$$\lambda' = (1 - h_p)(1 - h_{CS})\lambda \tag{3.1}$$

where h_p and h_{cs} are PIT and cache hit rate respectively.

Table 3.3: Notations and their meaning

Symbols	Meaning
N	Number of PIT entries in an NDN node
K	Number of reserved PIT entries for Prioritized Interests
i	Number of Prioritized entries in PIT, $0 \leq i \leq N$
j	Number of Non-Prioritized entries in PIT, $0 \leq j \leq N - K$
$P_{i,j}$	Probability of the system being in state (i, j), i.e., there are i Prioritized PIT entries and j Non-Prioritized PIT entries respectively.
λ_p	Prioritized Interest packet arrival rate.
μ_p	Prioritized Data packet service rate.
λ_{np}	Non-Prioritized Interest packet arrival rate.
μ_{np}	Non-Prioritized Data packet service rate.
$P_{block,p}$	Prioritized Interest Blocking probability.
$P_{block,np}$	Non-prioritized Interest Blocking probability.
P_{nft}	Non-prioritized forced termination probability.

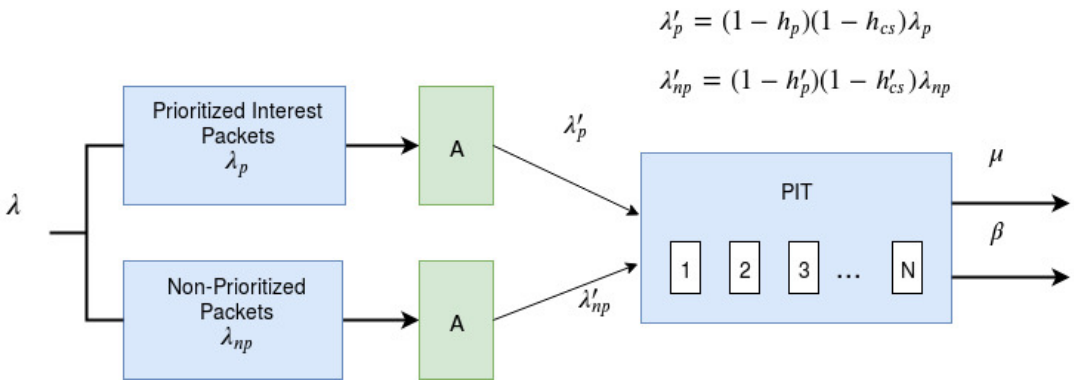


Figure 3.2: System Model for analytical modeling

Figure 3.2 shows the system model. The module A in the Figure 3.2 represents the Interest packet aggregation at PIT.

The rate of PIT entry removal due to time-out can be calculated as:

$$\beta = 1/\tau \tag{3.2}$$

where τ is the PIT entry time out (default value is 4 sec).

Since we consider service time for prioritized and non-prioritized Interests are exponentially distributed with μ_p and μ_{np}, the probability of service time for prioritized and non-prioritized Interests $\leq \tau$ can be calculated from cumulative distribution (c.d.f) formula.

The probability of Prioritized and Non-Prioritized Interests for which data packets arrive are calculated as below.

$\eta_p = 1 - e^{-\mu_p \tau}$

$\eta_{np} = 1 - e^{-\mu_{np} \tau}$

So, we can define $\mu'_p = \eta_p \mu_p + (1 - \eta_p)\beta$

$\mu'_{np} = \eta_{np} \mu_{np} + (1 - \eta_{np})\beta$

$\mu = \mu'_p + \mu'_{np}$

We use the following performance metrics:

1) *Prioritized Interest Blocking probability,* $P_{block,p}$: It is the probability that an incoming prioritized Interest arrives and finds the PIT full and is discarded.

2) *Non-prioritized Interest Blocking probability,* $P_{block,np}$: It is the probability that an incoming non-prioritized interest arrives and finds the PIT full and is discarded.

3) *Non-prioritized forced termination probability,* P_{nft}: It is the probability of removing non-prioritized PIT entries due to the arrival of a new prioritized Interest and PIT is full. This probability is more significant than others from the consumer's perspective. The higher this probability, the lower the consumer's ISR.

In order to obtain $P_{block,p}$, $P_{block,np}$ and P_{nft}, we calculate the steady-state probabilities of the state-transition diagram. In the following part, we have shown the evaluation of these probabilities for both schemes separately.

We calculate the steady-state probabilities $P_{i,j}$of the state-transition diagram to get $P_{block,p}$, $P_{block,np}$ and P_{nft}. The evaluation of these probabilities for each scheme (PRWR and PRR) are presented in the next section.

3.5.1 PRWR scheme

The PRWR scheme can be modelled as a 2-dimensional continuous-time markov process. The system states are denoted by (i, j) where i and j represent the number of prioritized and non-prioritized PIT entries, respectively. Figure 3.3 shows the transition rate diagram for PRWR scheme. For a better understanding of the transition rate diagram of the PRWR scheme, please refer to Figure 3.4. The state space is

$\Omega = \{(i, j)|0 \leq i \leq N, 0 \leq j \leq N\}$

In the following, we express equilibrium equations for each state of Figure 3.3.

i) $\boxed{\text{i=j=0}}$:$P_{i,j} * (\lambda'_{np}+\lambda'_p) = P_{i,j+1} * \mu'_{np} + P_{i+1,j} * \mu'_p$

ii) $\boxed{\text{1}\leq \text{j} \leq \text{N-1, i=0}}$: $P_{i,j} * (\lambda'_{np} + \lambda'_p + j * \mu'_{np}) = P_{i,j+1} * (j+1)\mu'_{np} + P_{i,j-1} * \lambda'_{np} + P_{i+1,j} * \mu'_p$

iii) $\boxed{\text{i=0, j= N}}$: $P_{i,j} * (j * \mu'_{np} + \lambda'_p) = P_{i,j-1} * \lambda'_{np}$

iv) $\boxed{\text{1}\leq \text{i} \leq \text{N-1 , j=0}}$: $P_{i,j} * (\lambda'_{np} + \lambda'_p + i * \mu'_p) = P_{i,j+1} * \mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (i+1)\mu'_p$

v) $\boxed{\text{i=N, j=0}}$: $P_{i,j} * (i * \mu'_p) = P_{i-1,j} * \lambda'_p + P_{i-1,j+1} * \lambda'_p$

vi) $\boxed{\text{1}\leq \text{i} \leq \text{N-2, 1}\leq \text{j} \leq \text{N-2, i+j}\leq\text{N-1}}$: $P_{i,j} * (j * \mu'_{np} + \lambda'_{np} + \lambda'_p + i * \mu'_p) = P_{i,j-1} * \lambda'_{np} + P_{i,j+1} * (j+1)\mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (i+1)\mu'_p$

vii) $\boxed{\text{1} \leq \text{i} \leq \text{N-1, i+j=N}}$: $P_{i,j} * (j * \mu'_{np} + i * \mu'_p + \lambda'_p) = P_{i,j-1} * \lambda'_{np} + P_{i-1,j} * \lambda'_p + P_{i-1,j+1} * \lambda'_p$

All the above equations (i)-(vii) can be combined and can be expressed as below.

$$\begin{aligned}
&\{i * \mu'_p + j * \mu'_{np} + [1 - \delta(N - i - j)] * \lambda'_{np} + [1 - \delta(N - i)] * \lambda'_p\} * P_{i,j} = \\
&(i+1) * \mu'_p * [1 - \delta(N - i - j)] * P_{i+1,j} + (j+1) * \mu'_{np} * [1 - \delta(N - i - j)] * P_{i,j+1} + \\
&\lambda'_{np} * [1 - \delta(j)] * P_{i,j-1} + \lambda'_p * [1 - \delta(i)] * P_{i-1,j} + \lambda'_p * [1 - \delta(i)] * \delta(N - i - j) * P_{i-1,j+1}
\end{aligned} \tag{3.3}$$

where $\delta(k)$ is a step function. k is a linear function of i, j and N and $i + j \leq N$.

$$\delta(k) = \begin{cases} 1 & k = 0 \\ 0, & \text{otherwise} \end{cases}$$

For normalizing the steady-state probabilities,

$$\sum_{i}\sum_{j} P_{i,j} = 1 \qquad \forall i+j \leq N \tag{3.4}$$

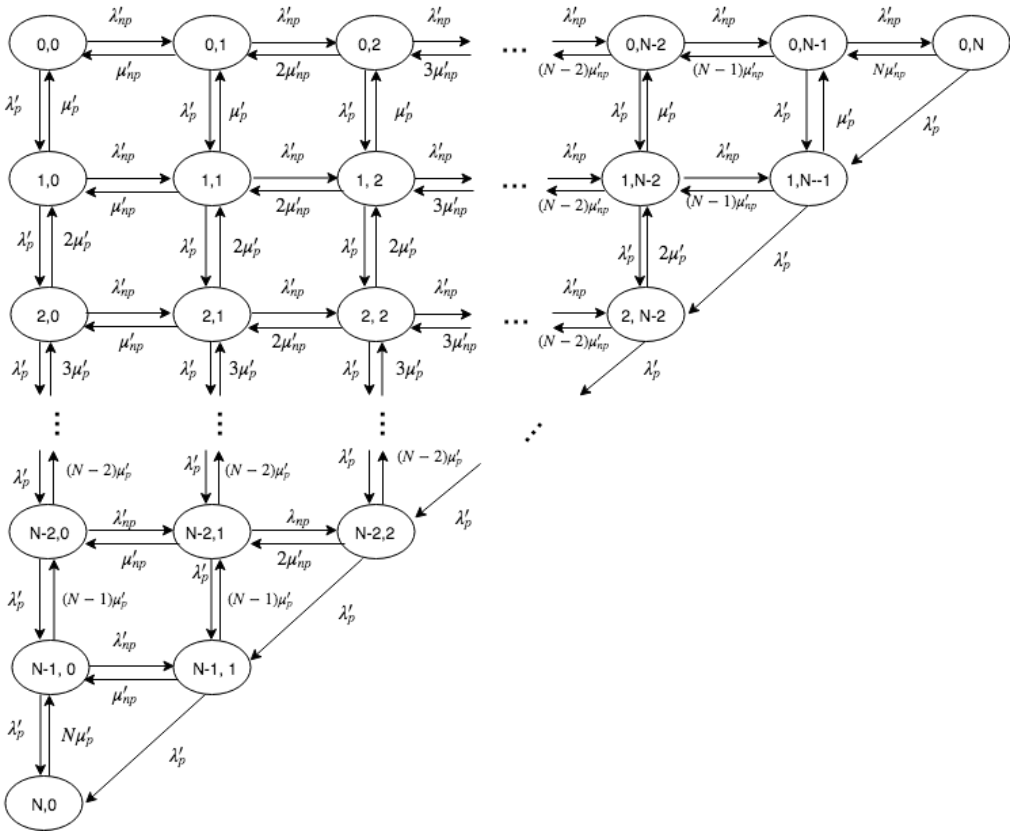


Figure 3.3: Transition rate diagram for PRWR scheme

Performance metrics

(i) *Prioritized Interest Blocking probability,* $P_{block,p}$: It is given by the steady-state probability of state (N,0) as

$P_{block,p} = P_{N,0}$

The value of $P_{N,0}$ can be calculated using the Erlang B formula.

$$P_{N,0} = \frac{\frac{A_p^n}{n!}}{\sum_{l=0}^{N} \frac{A_p^l}{l!}} \tag{3.5}$$

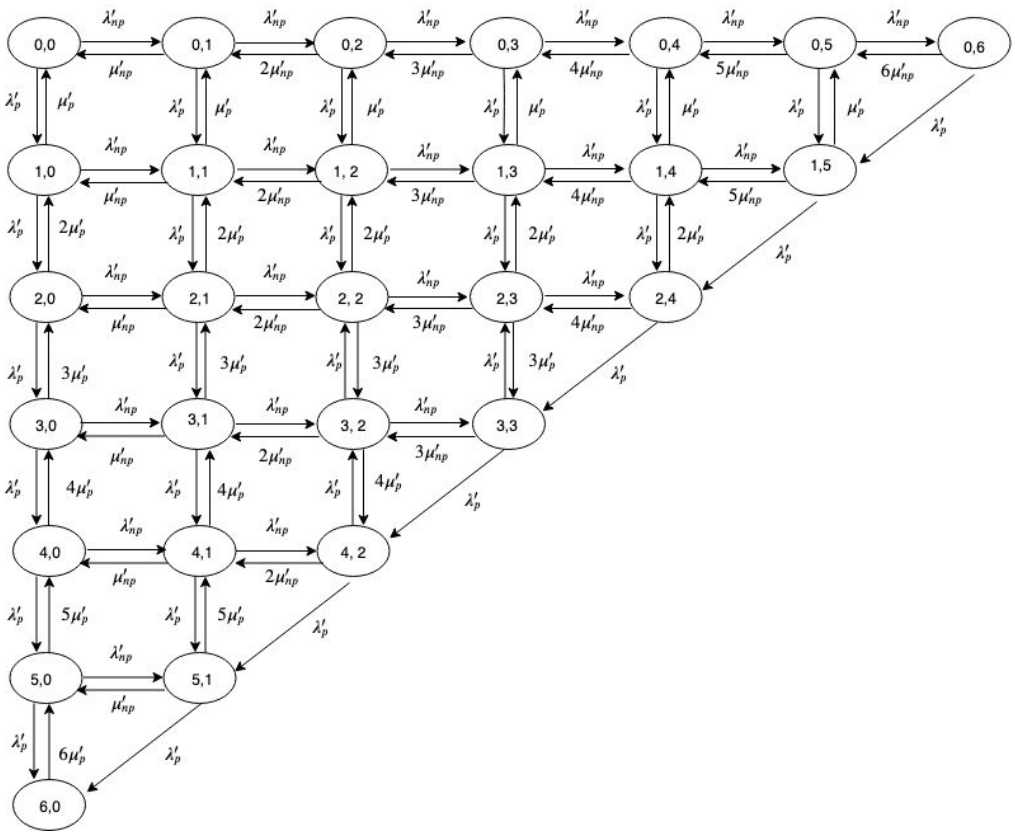


Figure 3.4: Transition rate diagram for PRWR scheme considering N=6

ii) *Non-prioritized blocking probability*, $P_{block,np}$: It is expressed as

$$P_{block,np} = \sum_{i,j;i+j=N} P_{i,j} \tag{3.6}$$

iii) *Non-prioritized forced termination probability*, P_{nft}: It is expressed as

$$P_{block,np} = \sum_{i,j;i+j=N} P_{i,j} * \lambda_p/\lambda_{np} \tag{3.7}$$

The reason behind the Equation (3.7) is as follows:

The average time for which an entry stays in state (i, j) for time duration td=$P(i, j) * td$. So, the average number of replaced non-prioritized PIT entries= $P(i, j) * td * (j/N - i)\lambda_p$.

The average number of non-prioritized interest arrival for time duration td is $\lambda_{np} * td$. Now,

we can specify P_{nft} as a function of the number of replaced non-prioritized PIT entries over the number of non-prioritized interest packet arrivals in period ts.

In PRWR scheme, non-prioritized entries are replaced when $i+j = N$ (PIT full), so Equation (3.7) becomes the final expression.

3.5.2 PRR scheme

The PRR scheme can be modeled as 2-dimensional continuous-time markov process. Figure 3.5 shows the Transition rate diagram of PRR scheme. For better understanding of the transition rate diagram of PRR scheme, please refer to Figure 3.6. The state space is defined as:

$\Omega = \{(i, j)|0 \leq i \leq N, 0 \leq j \leq N\}$

where i and j are the numbers of prioritized and non-prioritized PIT entries, respectively.

In the following, we express equilibrium equations for each state of Figure 3.5.

i) $\boxed{\text{i=j=0}}$: $P_{i,j} * (\lambda'_{np} + \lambda'_p) = P_{i,j+1} * \mu'_{np} + P_{i+1,j} * \mu'_p$

ii) $\boxed{\text{1}\leq \text{j} \leq \text{N-K-1, i=0}}$: $P_{i,j} * (\lambda'_{np} + \lambda'_p + \text{j} * \mu'_{np}) = P_{i,j+1} * (\text{j+1})\ \mu'_{np} + P_{i,j-1} * \lambda'_{np} + P_{i+1,j} * \mu'_p$

iii) $\boxed{\text{i=0, j= N-K}}$: $P_{i,j} * (\ \text{j} * \mu'_{np} + \lambda'_p) = P_{i,j-1} * \lambda'_{np} + P_{i+1,j} * \mu'_p$

iv) $\boxed{\text{1}\leq \text{i} \leq \text{N-K-1 , j=0}}$: $P_{i,j} * (\lambda'_{np} + \lambda'_p + \text{i} * \mu'_p) = P_{i,j+1} * \mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\mu'_p$

v) $\boxed{\text{N-K}\leq \text{i} \leq \text{N-1 , j=0}}$: $P_{i,j} * (\lambda'_p + \text{i} * \mu'_p) = P_{i,j+1} * \mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\mu'_p$

vi) $\boxed{\text{i=N, j=0}}$: $P_{i,j} * (\text{i}*\mu'_p) = P_{i-1,j} * \lambda'_p + P_{i-1,j+1} * \lambda'_p$

vii) $\boxed{\text{1}\leq \text{i} \leq \text{N-K-2, 1}\leq \text{j} \leq \text{N-K-2, i+j}\leq\text{N-K-1}}$: $P_{i,j} * (\text{j}*\mu'_{np} + \lambda'_{np} + \lambda'_p + \text{i} * \mu'_p) = P_{i,j-1} * \lambda'_{np} + P_{i,j+1} * (\text{j+1})\mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\ \mu'_p$

viii) $\boxed{\text{1}\leq \text{i} \leq \text{K-1, j= N-K}}$: $P_{i,j} * (\text{j}* \mu'_{np} + \text{i} * \mu'_p + \lambda'_p) = P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\ \mu'_p$

ix) $\boxed{\text{i=K, j= N-K}}$: $P_{i,j} * (\text{j}* \mu'_{np} + \text{i} * \mu'_p + \lambda'_p) = P_{i-1,j} * \lambda'_p$

x) $\boxed{\text{K+1}\leq \text{i} \leq \text{N-1, 1}\leq \text{j} \leq \text{N-K-1, i+j=N}}$: $P_{i,j} * (\text{j}* \mu'_{np} + \text{i} * \mu'_p + \lambda'_p) = P_{i-1,j} * \lambda'_p + P_{i-1,j+1} * \lambda'_p$

xi) $\boxed{\text{K}\leq \text{i} \leq \text{N-2, 1}\leq \text{j} \leq \text{N-K-1, i+j}\leq\text{N-1}}$: $P_{i,j} * (\text{j}* \mu'_{np} + \text{i} * \mu'_p + \lambda'_p) = P_{i,j} * (\text{j}* \mu'_{np} + \lambda'_{np} + \lambda'_p + \text{i} * \mu'_p) = P_{i,j+1} * (\text{j+1})\ \mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\ \mu'_p$

xii) $\boxed{\text{1}\leq \text{i} \leq \text{N-K-1, 1}\leq \text{j} \leq \text{N-K-1, i+j}\leq\text{N-K}}$: $P_{i,j} * (\text{j}* \mu'_{np} + \lambda'_p + \text{i} * \mu'_p) = P_{i,j-1} * \lambda'_{np} + P_{i,j+1} * (\text{j+1})\ \mu'_{np} + P_{i-1,j} * \lambda'_p + P_{i+1,j} * (\text{i+1})\mu'_p$

All the above equations (i)-(xii) can be combined and can be expressed as below:

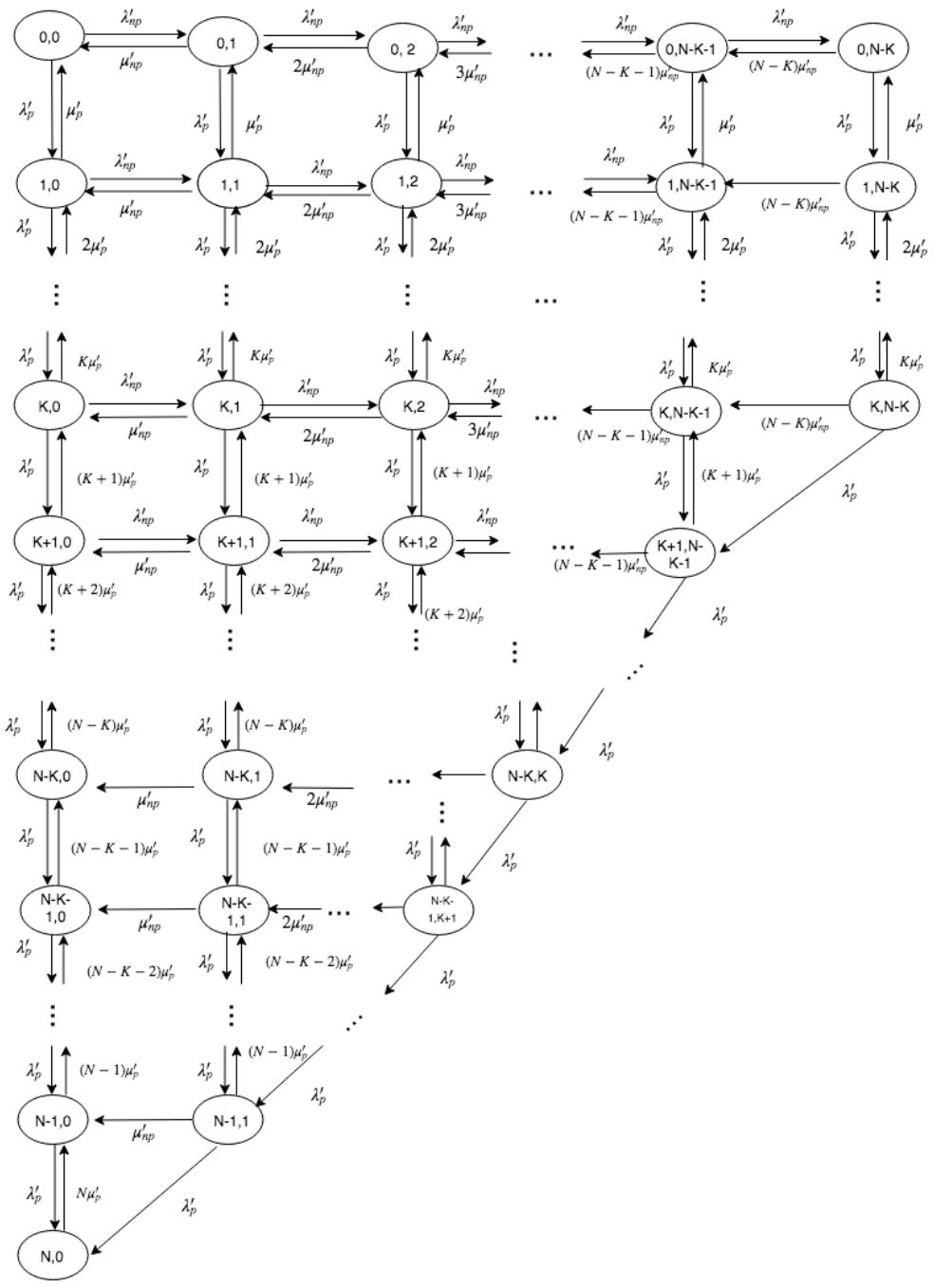


Figure 3.5: Transition rate diagram for PRR scheme

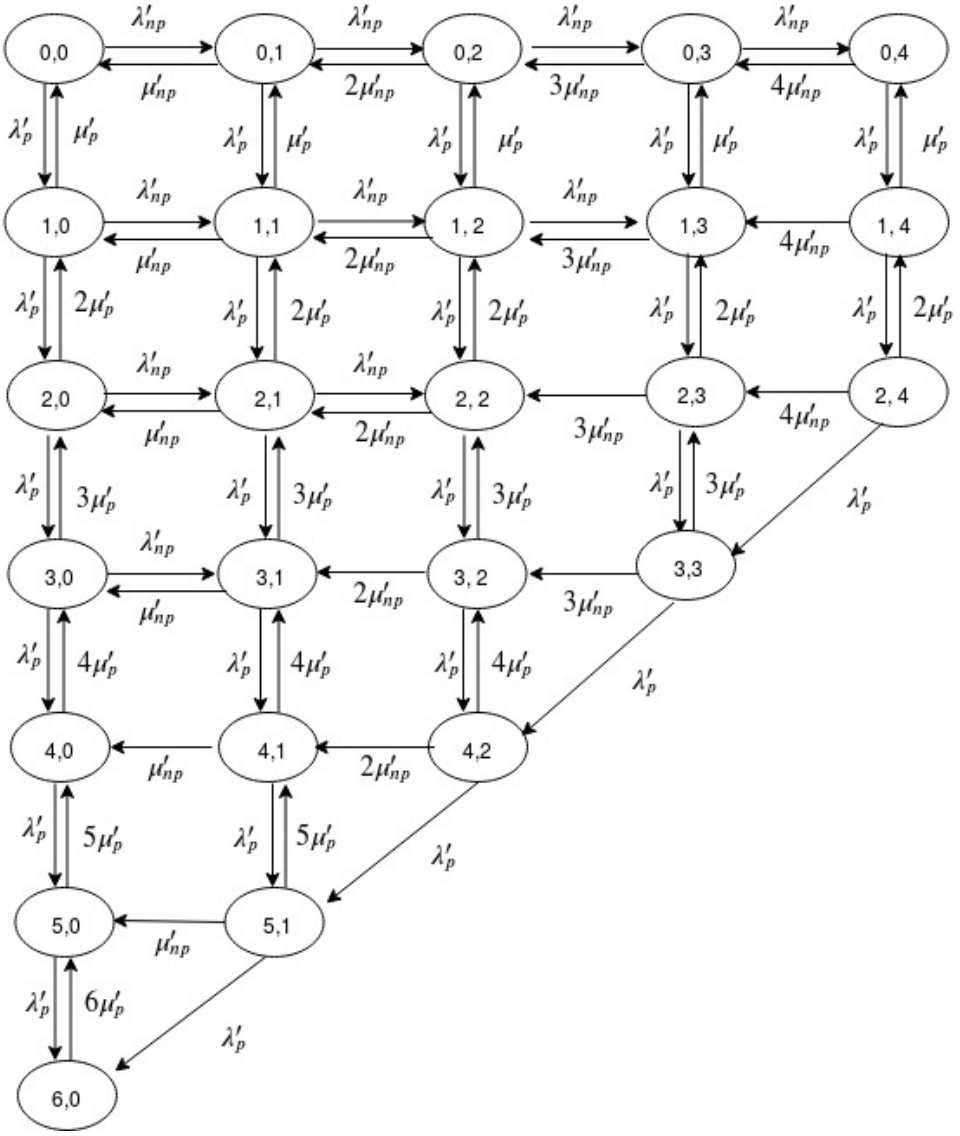


Figure 3.6: Transition rate diagram for PRR scheme considering N=6, K=2

The balance equations for the PRR scheme are shown below.

$$\{q * \lambda'_{np} + q_1 * \lambda'_p + i * \mu'_p + j * \mu'_{np}\} * P_{i,j} = q_2 * P_{i-1,j} * \lambda'_p + q_3 * P_{i+1,j} * (i+1)\mu'_p + q_4 * P_{i,j+1} * (j+1) * \mu'_{np} + q_5 * j * P_{i,j-1} * \lambda'_{np} + q_6 * P_{i-1,j+1} * \lambda'_p \quad (3.8)$$

where

$$q = \begin{cases} 0 & N-K \leq i+j \leq N \\ 1 & otherwise \end{cases}$$

$$q1 = \begin{cases} 0 & i = N \\ 1 & otherwise \end{cases}$$

$$q2 = \begin{cases} 1 & i > 0 \\ 0 & otherwise \end{cases}$$

$$q3 = \begin{cases} 0 & i+j = N \\ 1 & otherwise \end{cases}$$

$$q4 = \begin{cases} 0 & j \geq N-K \quad or \\ & i+j = N \\ 1 & otherwise \end{cases}$$

$$q5 = \begin{cases} 0 & \text{i+j ¿ N-K} \quad or \\ & i+j = N \\ 1 & otherwise \end{cases}$$

$$q6 = \begin{cases} 1 & i+j = N \quad and \\ & j! = N-K \\ 0 & otherwise \end{cases}$$

The sum of all the state probabilities is equal to 1 as below

$$\sum_i \sum_j P_{i,j} = 1 \qquad \forall i+j \leq N \tag{3.9}$$

Performance metrics

i) *Non-prioritized blocking probability*, $P_{block,np}$: A non-prioritized interest is blocked at the states where $i + j \geq N - K$. It is expressed as

$$P_{block,np} = \sum_{i,j;i+j \geq N-K} P_{i,j} \tag{3.10}$$

The Prioritized Interest Blocking probability and Non-prioritized forced termination probability can be similarly obtained using Equations (3.5) and (3.7) respectively.

3.6 Performance Evaluation

To ensure the correctness of our markovian model and assumptions, we compare the analytical results with the simulation results. The values of N, λ'_p, $\lambda_{np'}$, μ'_p and $\mu_{np'}$ are required for numerical calculation. From the simulation, we collect these values.

We implement our proposed schemes: PRWR and PRR in the ndnSIM simulator (ndnSIM 2.4) [7]. The configuration of the system that we used for the simulation experiment is as follows: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 16GB RAM and Ubuntu 16.04 LTS as the operating system. We run the simulation for 100 seconds. We set the Interest lifetime to their default value (4 sec). To overlook the influence of in-network caching, ConsumerCbr application is installed in all consumers. This application sends Interests at a constant average rate.

For better understanding, we first consider two simple topologies (refer to Figure 3.7).

1. *bottleneck topology (6-nodes)*: number of consumers=2, number of producers=2, link bandwidth=10 Mbps except bottleneck link (R1-R2) which is of 1 Mbps bandwidth, link delay=10ms.

2. *ndn-grid topology (9-nodes)*: number of consumers=1, number of producers=1, Link bandwidth=10 Mbps, link delay=10ms.

Later, we use a realistic, large-scale network topology: *Rocketfuel topology (176-nodes)* to verify the results from smaller topologies. Please refer to Figure 3.8. In the Rocketfuel topology, there are three types of nodes: (i) nodes with degree < 4 are named as clients(130 red nodes), (ii) nodes directly connected to clients are termed as gateway (33 green nodes), and (ii) remaining nodes as backbone (13 blue nodes). We assign random numbers (within some ranges) to link

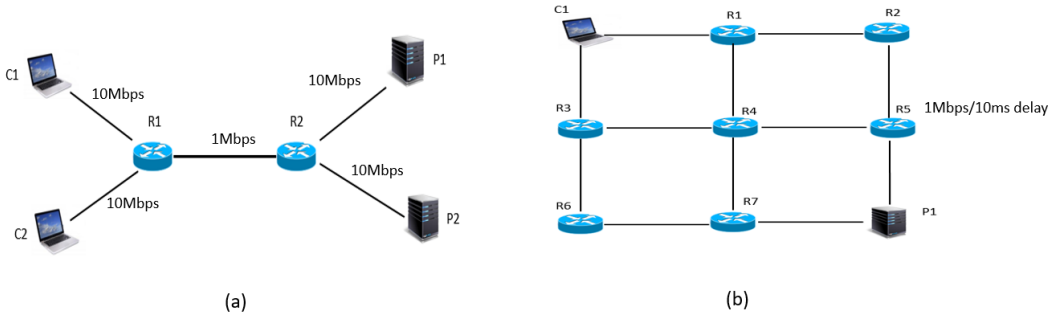


Figure 3.7: (a) bottleneck topology (b) ndn-grid topology

Bandwidth and delay as shown in Table 3.4. We installed the data producer in the backbone nodes in the simulation experiment.

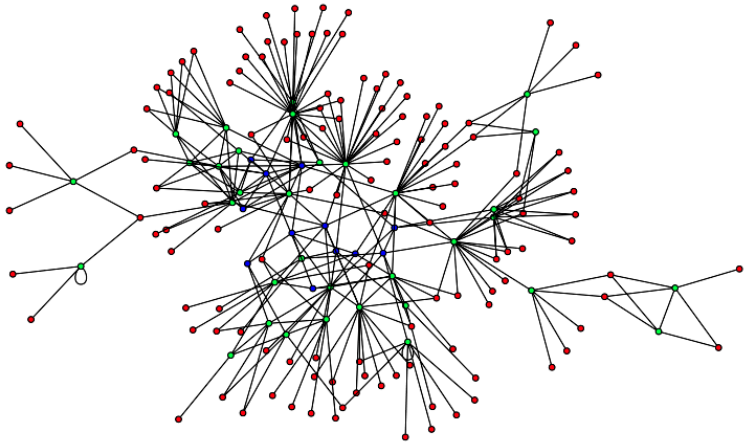


Figure 3.8: Rocketfuel topology (130 client routers (red), 33 gateway routers(green), 13 backbone routers (blue))

The simulation parameters are listed in Table 4.2.

The first challenge we face in the experimental setup is determining the PIT size. Over-allocation of memory to PIT does not improve network performance (discussed in section 3.1). The simulation results are presented based on 20 runs per experiment and with a 95% confidence interval. We calculate the best possible PIT size from simulation, which we discuss in the following subsection.

Table 3.4: Link Bandwidth and Delay Ranges in Rocketfuel topology

Link Type	**Delay**		**Bandwidth**	
	Min	**Max**	**Min**	**Max**
Client -Gateway	10ms	70ms	1Mbps	3Mbps
Backbone -Backbone	5ms	10ms	40Mbps	100Mbps
Gateway -Backbone	5ms	10ms	40Mbps	100Mbps
Gateway -Gateway	5ms	10ms	10Mbps	20Mbps

Table 3.5: Simulation parameters in ndnSIM simulator

Parameter	*Value*
Forwarding Strategy	Best-Route Strategy
Interest lifetime	4000 ms
Interest Packet Size	215 Byte
Data Size	1054 Byte
% of Prioritized Interest	40%
Zipf Factor α	0.7
Simulation Time	100 sec

3.6.1 PIT size estimation

In theory, we can use Little's Law to calculate the PIT size of a node.

$$PS = (1 - h_p)(1 - h_{CS})\lambda * T \tag{3.11}$$

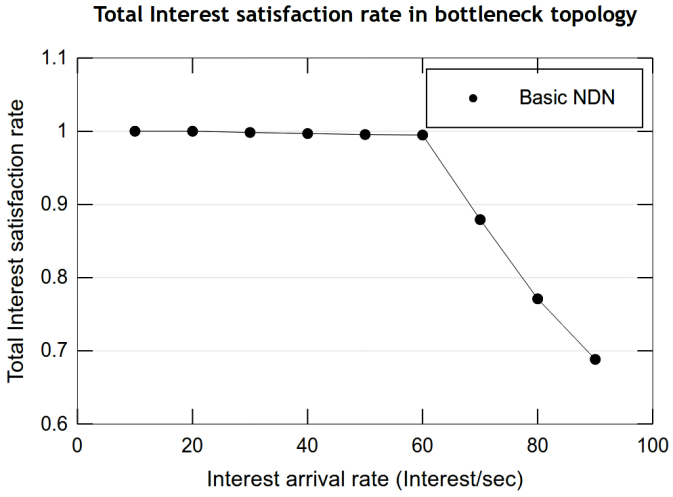
where PS is PIT size in terms of number of entries, h_p is PIT hit rate, h_{CS} is CS hit rate, λ is the Interest arrival rate, and T is the average response time of each PIT entry.

We do not have knowledge of the Interest arrival rate at each node ahead of simulation. Only the arrival rates at each consumer node can be changed. As a result, calculating PIT size from Equation (3.11) is difficult. We observe the relation between total satisfaction rate and average response time at different Interest arrival rates. From the relation, we try to fix the PIT size.

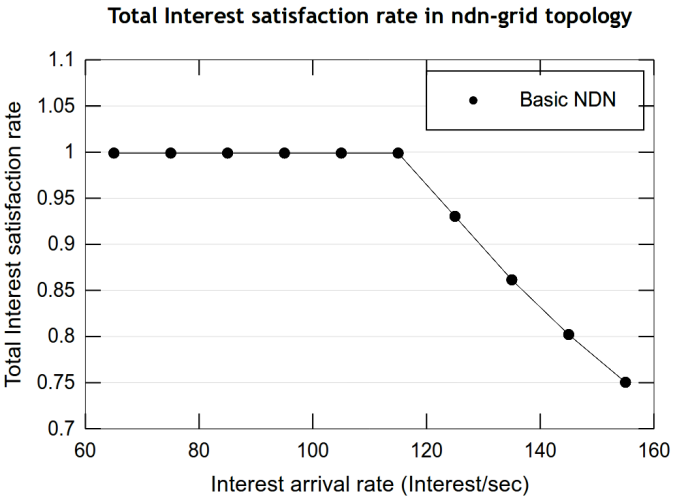
The total satisfaction rate is calculated as follows:

$$ISR = \frac{D}{I} \tag{3.12}$$

where ISR is total Interest satisfaction rate, D is the total number of received Data packets by all consumers and I is total Interest packets by generated all consumers.

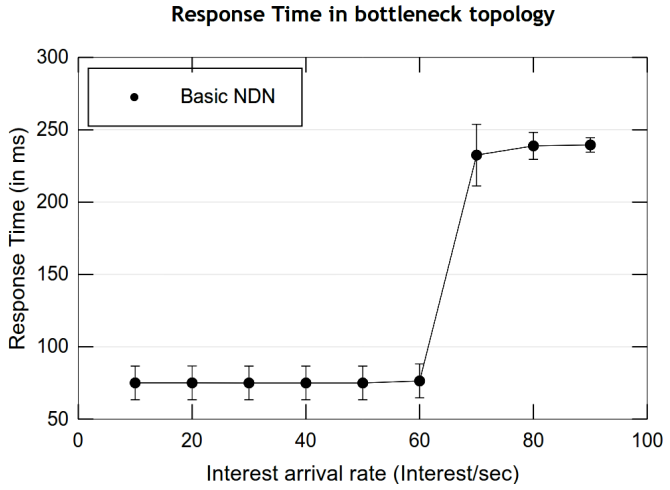


(a)

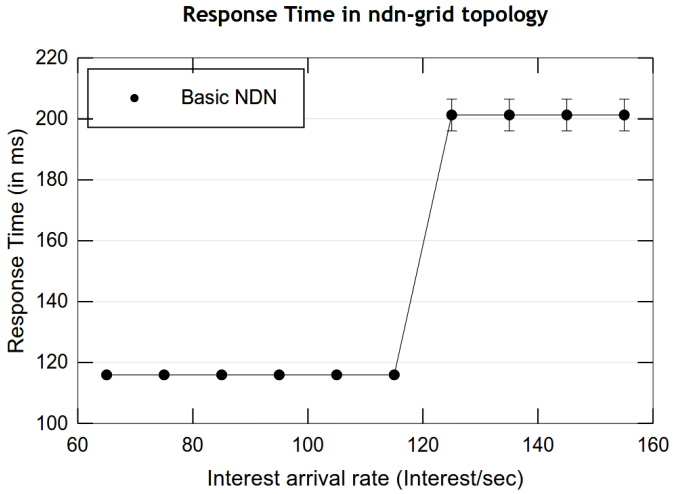


(b)

Figure 3.9: Total Interest Satisfaction Rate vs Interest Frequency in (a) bottleneck topology and (b) ndn-grid topology.



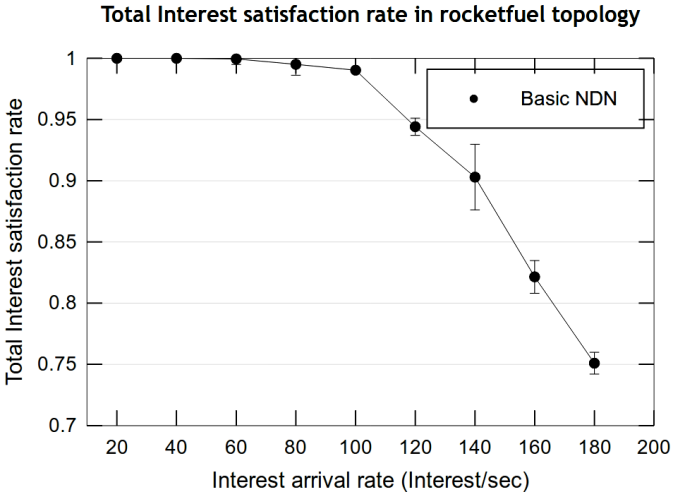
(a)



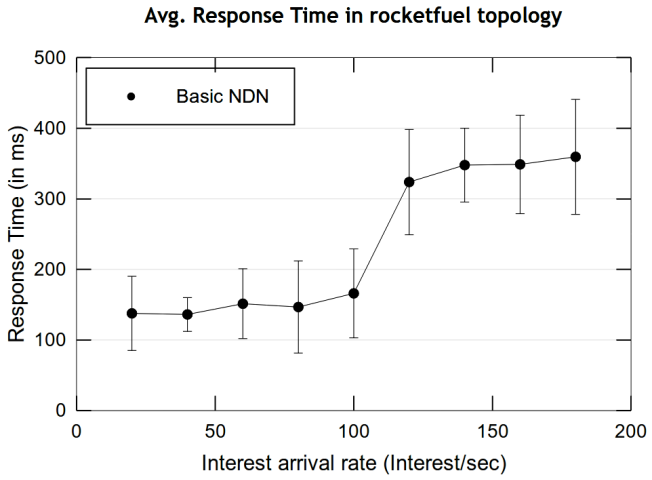
(b)

Figure 3.10: Average Response/service time of a consumer vs Interest arrival rate in (a) bottleneck topology and (b) ndn-grid topology. The error bars in the plot indicate 95% confidence intervals.

As seen in Figures 3.9(a), Fig. 3.9(b) and Fig. 3.11(a), the total satisfaction rate initially reduces modestly with increasing arrival rate and then rapidly declines at a specific Interest arrival rate (IAR). Please note that we have not limited the PIT size in this simulation. From



(a)



(b)

Figure 3.11: (a) Total Interest satisfaction rate vs Interest arrival rate (b) Average Response/service time of a consumer vs Interest arrival rate in rocketfuel topology. The error bars in the plot indicate 95% confidence intervals..

the result, we found the *IAR* value equal to 60, 115 and 100 interests/sec for bottleneck, ndn-grid and rocketfuel topology, respectively (refer to Table 3.6). Furthermore, we observe a sharp increase in average response time at the same frequencies before (refer to Figure 3.10(a), 3.10(b)

and 3.11(b)).

During the simulation, we count the maximum number of PIT entries in each node to determine the PIT size. We consider the PIT size as the maximum value of all nodes at IAR=60, 115 and 100 Interests/sec for bottleneck, ndn-grid and rocketfuel topology, respectively. Figure 3.6 shows the Interest Arrival Rate and the corresponding Maximum PIT size for bottleneck, ndn-grid and rocketfuel topology.

Table 3.6: Maximum PIT size for different topologies

Topology	Interest Arrival Rate	Max PIT size
Bottleneck	60	21
ndn-grid	115	25
rocketfuel	100	53

3.6.2 Comparison of PRR and PRWR scheme with Basic NDN, RAPIT [4]

To compare our proposed schemes with Basic NDN and RAPIT [4], we implement RAPIT in the ndnSIM v2.4 simulator. RAPIT [4] is designed for use in a network where packet loss occurs frequently. They took a 2% packet loss rate into account (we implement it in the simulator by dropping packets at producers at a rate of 2%). The main aim of RAPIT [4] is to shorten the residence time for non-responded entries in PIT. As a result, each router in NDN adjusts its residence time dynamically based on the estimated RTT. RTT is calculated between the content publisher and itself. After each $Tmsr$ time duration, RTT is estimated. The decision of replacement is considered depending on a parameter named *importance*. When PIT becomes full, the router compares the important value of existing PIT entries and the new incoming Interest. The one having the lowest among them is selected for eviction. For an incoming Interest, depending on the network condtion, the *importance* value is set. For good network condtion, it is 1 otherwise δ (= 1/3). However, *importance* value for existing PIT entries depends on η value. η is calculated as: η = time left for PIT entry to expire/ Pending Entry Lifetime. If $\eta \leq \delta$, then importance of the entry is η, otherwise 1.

Please note that while we are comparing RAPIT to Basic NDN, PRR scheme, and PRWR scheme, we have not included packet loss rate in the simulation. RAPIT is also not designed

with quality of service in mind.

We can observe from the simulation results (refer to Figures 3.12 to 3.16) that Basic NDN outperforms RAPIT on all of the performance criteria listed below. We discovered the following likely reasons as a result of our investigation:

- We learned from the experiment that when PIT fills up, an existing PIT entry is replaced each time. The reason for this is that there is always an existing PIT entry with a lower *importance* than the new interest (lowest *importance* value of an incoming Interest in RAPIT is .33). Customers are less satisfied when existing entries are replaced frequently.

- Furthermore, we discovered that the performance of RAPIT is significantly dependent on the RTT calculation interval $Tmsr$. With the increase in time interval T_{msr}, the interest satisfaction rate declines. It happens as we dynamically set the PIT residency time. Before the data packet arrives, the PIT entries are removed. The response time increases faster as the interest arrival rate increases, whereas the residence time is set based on the previously estimated RTT value. In the simulation, T_{msr} is equal to 1 sec.

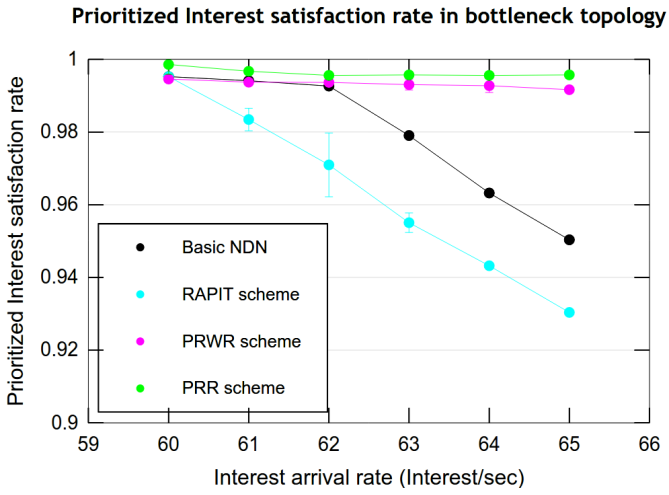
i) *Prioritized Interest satisfaction rate:*

The prioritised Interest satisfaction rate is calculated as follows,

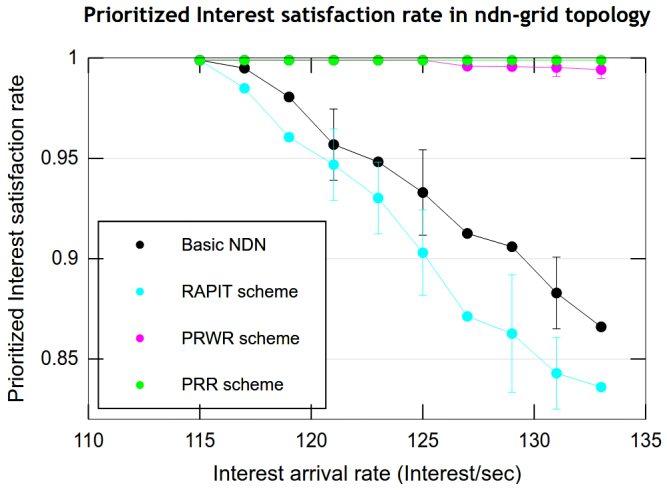
$$PISR = \frac{PD}{PI} \tag{3.13}$$

where $PISR$ is the prioritized Interest satisfaction rate, PD is the total number of Data packets received by all consumers corresponding to the prioritized Interest packet sent, and PI is the total number of prioritized Interest packets sent by all consumers.

From Figures (3.12(a)), (3.12(b)) and (3.15(a)), we observe that both PRWR and PRR achieve a significant increase in performance in terms of prioritized interest satisfaction rate (99%). The reasons for not achieving 100% are: (1) Some interests are on the fly due to the measurement window (2) Some PIT entries expire due to queuing delay. Furthermore, while comparing PRR to PRWR, we observe a slight increase in the prioritized interest satisfaction rate. The reason for this is because, in PRR, some non-prioritized packets are discarded early due to reservation; in PRWR, on the other hand, the data packets corresponding to replacement entries consume traffic and cause slight congestion, causing more prioritised packets to time out than in PRR.



(a)



(b)

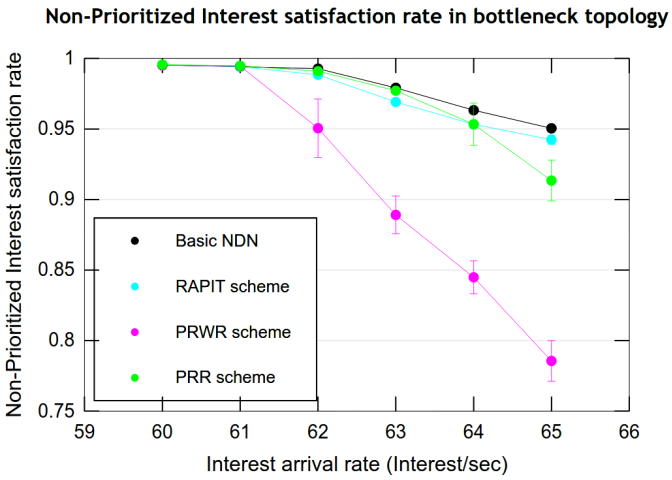
Figure 3.12: Prioritized Interest satisfaction rate vs Interest arrival rate (a) bottleneck topology (b) ndn-grid topology. The error bars in the plot indicate 95% confidence intervals.

ii) *Non-Prioritized Interest satisfaction rate:*

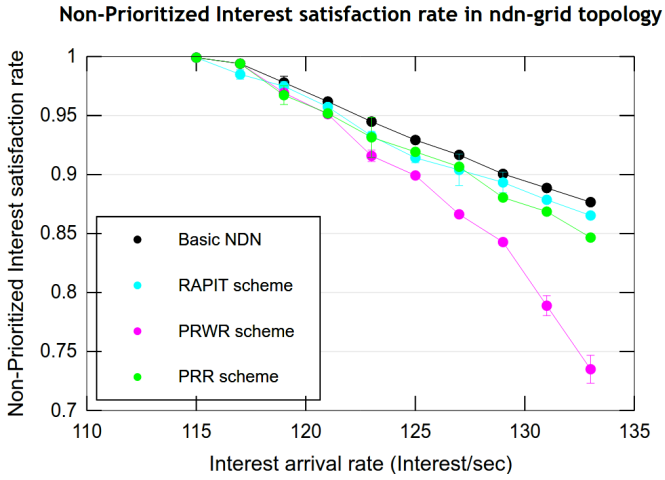
The non-prioritised Interest satisfaction rate is calculated as follows,

$$NPISR = \frac{NPD}{NPI} \tag{3.14}$$

where $NPISR$ is the non-prioritized Interest satisfaction rate, NPD is the total number of



(a)

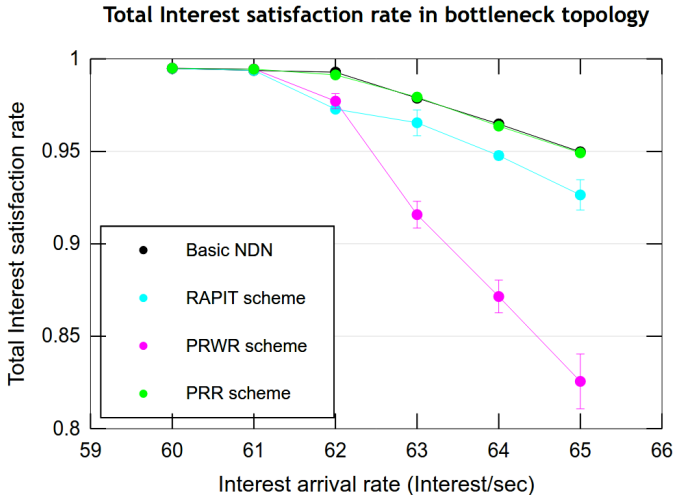


(b)

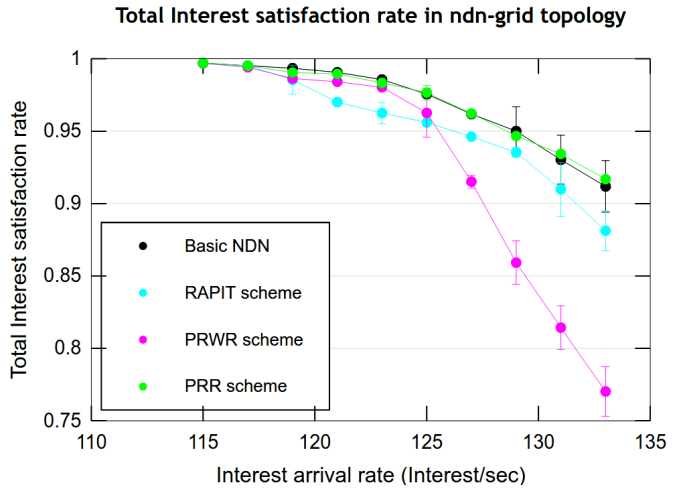
Figure 3.13: Non-Prioritized Interest satisfaction rate vs Interest arrival rate (a) bottleneck topology (b) ndn-grid topology. The error bars in the plot indicate 95% confidence intervals.

Data packets received by all consumers corresponding to non-prioritized Interest packets sent and NPI is the total number of non-prioritized Interest packets sent by all consumers.

As shown in Figures (3.13(a)), (3.13(b)) and (3.15(b)), the non-prioritized interest satisfaction rate degrades in both PRR and PRWR scheme with the increase in Interest arrival rate, as compared to Basic NDN. This happens due to the eviction of non-prioritized PIT entries to



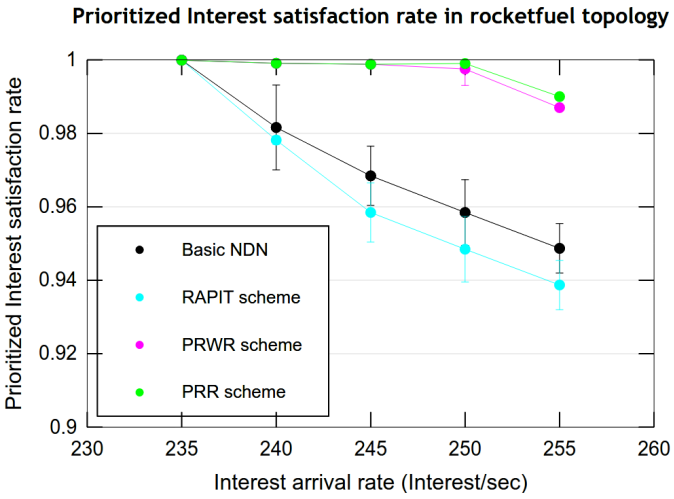
(a)



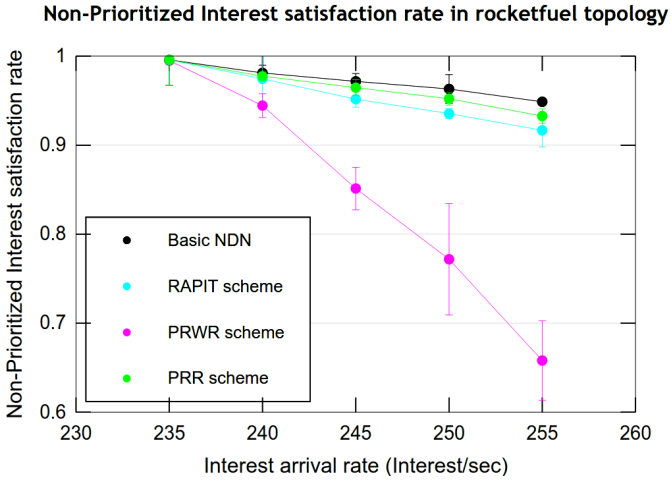
(b)

Figure 3.14: Total Interest satisfaction rate vs Interest arrival rate (a) bottleneck topology (b) ndn-grid topology. The error bars in the plot indicate 95% confidence intervals.

make room for prioritized Interest packets when PIT is full. Furthermore, we find that PRR has a higher non-prioritized Interest satisfaction rate than PRWR. This is because the amount of non-prioritized entries evicted in PRR is lower due to the reservation for Prioritized entries.



(a)



(b)

Figure 3.15: (a) Prioritized Interest satisfaction rate vs Interest arrival rate (b) Non-prioritized Interest satisfaction rate vs Interest arrival rate in rocketfuel topology. The error bars in the plot indicate 95% confidence intervals.

iii) *Total Interest satisfaction rate:*

We use Equation (3.12) to compute the total Interest satisfaction rate. We observe that both PRR and Basic NDN provide similar total Interest satisfaction rate (refer to Figures 3.14(a), 3.14(b) and 3.16). This is possible as we can satisfy prioritized interests in lieu of non-prioritized

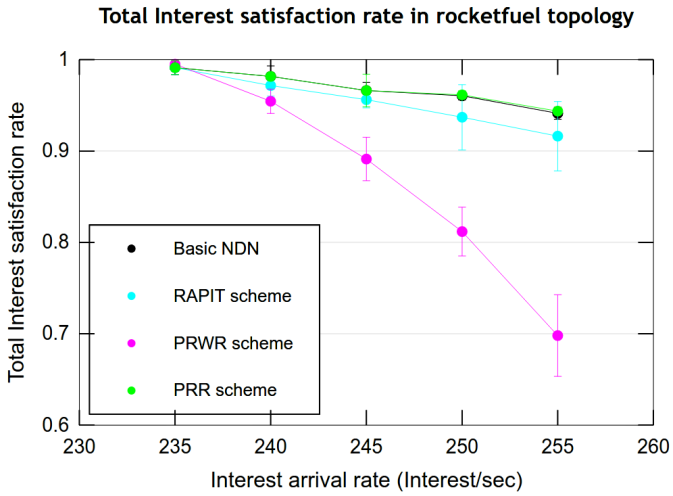


Figure 3.16: Total Interest satisfaction rate vs Interest arrival rate in rocketfuel topology. The error bars in the plot indicate 95% confidence intervals

Interest packet drop.

Compared to Basic NDN and PRR, the PRWR shows a lower total Interest satisfaction rate. The reason for this is that when a Prioritized Interest arrives and finds the PIT is filled up, we replace non-prioritized entries in PRWR. In the case of PRR, the reservation for prioritised Interests helps to reduce the number of non-prioritized entries that need to be replaced. There is no reservation in the PRWR scheme, resulting in more non-prioritized entries being replaced. As a result, Data packets matching those deleted PIT entries are unable to be sent downstream, resulting in a lower overall Interest satisfaction rate.

3.6.3 Model validation

To calculate the steady-state probability of each state, we solve the linear equations. For PRWR, we solve Equations (3.3)-(3.4) and for PRR, we solve Equations (3.8)-(3.9) by utilizing Bicon-jugate gradient method using pre-conditioner. We chose this method because it works for non-symmetric matrices. We use Intel Math Kernel Library (MKL) to speed up the calculation. We investigate the trade-off between various parameters at varying traffic loads using analytical and simulated results.

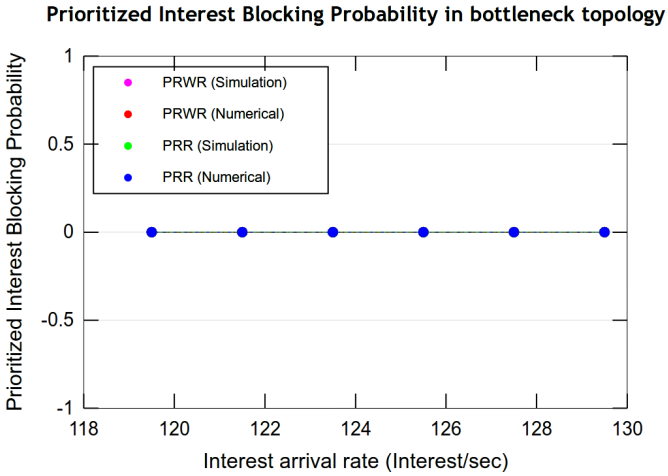
We derive the simulation results for the average blocking and forced termination probabilities with a 95% confidence interval. By taking a different percentage of priority Interests, we can

investigate these probabilities further. We consider 30% and 40% of prioritized Interests. In this subsection, we evaluate and compare different performance metrics for both PRWR and PRR schemes. All nodes in the topology have a fixed PIT size. We validate the model for node R1 in bottleneck and ndn-grid topology (refer to Figure 3.7). In the case of rocketfuel topology (refer to Figure 3.8), we consider the node with the largest PIT size (while running Basic NDN).

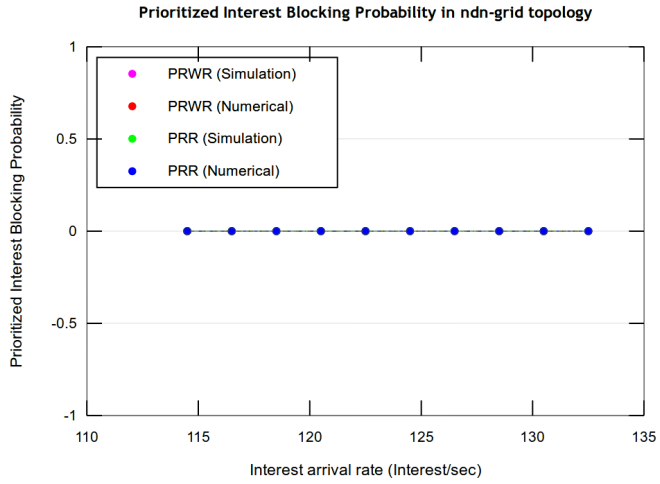
We observe that the prioritized Interest blocking probability $P_{block,p}$ for both PRWR and PRR schemes is zero (refer to Figures 3.17(a), 3.17(b) and 3.18). It is because prioritized Interest packets are given preference to get an entry into PIT over others. It's worth noting that we considered 30% and 40% of the interest packets as prioritized interests.

The comparison of non-prioritized Interest Blocking probability ($P_{block,np}$) for PRWR and PPR scheme w.r.t Interest arrival rate in bottleneck, ndn-grid, and rocketfuel topology is shown in Figures 3.19, 3.20 and 3.23. As shown in Figures 3.19, 3.20 and 3.23, the blocking probability for the PRR scheme increases as the interest arrival rate increases, whereas the blocking probability for the PRWR system decreases. It happens due to the application of dynamic reservation for prioritized interests in the PRR scheme. As a result, when a particular number of PIT entries are set aside for prioritised interests, non-prioritized interests have fewer options, resulting in more $P_{block,np}$. Moreover, we also observe that with the increase of prioritized Interests, blocking probability increases for both PRWR and PRR schemes. The reason behind it is that increased prioritized Interests help to increase the number of blocked non-prioritized Interests. Furthermore, we observe that when the number of prioritised Interests increases, the probability of blocking increases for both PRWR and PRR schemes. The rationale behind this is that having more prioritised Interests allows us to have more blocked non-prioritized Interests.

The non-prioritized forced termination probability for both PRWR and PPR schemes in bottleneck, ndn-grid, and rocketfuel topologies is shown in Figures 3.21, 3.22 and 3.24 respectively. From the figures, we observe that with the PRWR scheme, with the increase in Interest arrival rate, P_{nft} increases. It is because when a prioritised Interest arrives and PIT is already at maximum capacity, existing non-prioritized entries are replaced. As a result, higher the λ_{np}, non-prioritized items are more likely to be replaced or forcibly terminated, resulting in a higher P_{nft}. However, in the PRR scheme, even if P_{nft} increases as the arrival rate increases due to dynamic reservation, P_{nft} remains constant.



(a)



(b)

Figure 3.17: Prioritized Interest Blocking probabilities (a) bottleneck topology (b) ndn-grid topology. The error bars in the plot indicate 95% confidence intervals. The simulation results are same both for 30% and 40% prioritized Interests.

We can see from the experiment that the higher the value of K (reserved entries) for any λ' leads to lower P_{nft} and higher $P_{block,np}$ for PRR scheme. As a result, we calculate the value of K dynamically to keep the P_{ft} within the acceptable range. The value of tolerable non-prioritized forced termination probability is statically set to 0.01 in the simulation.

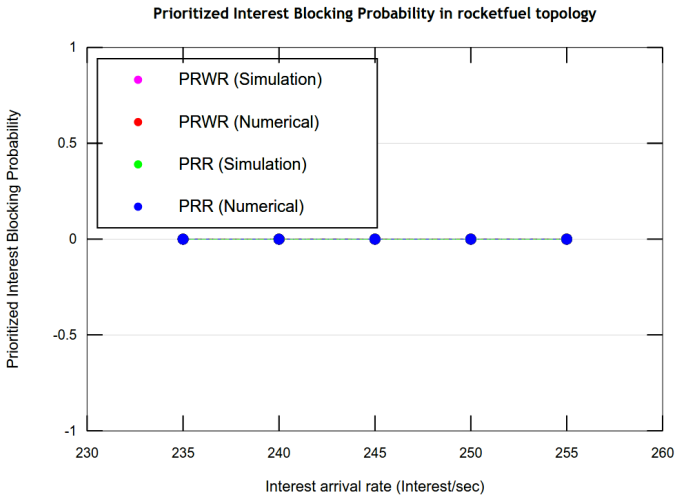
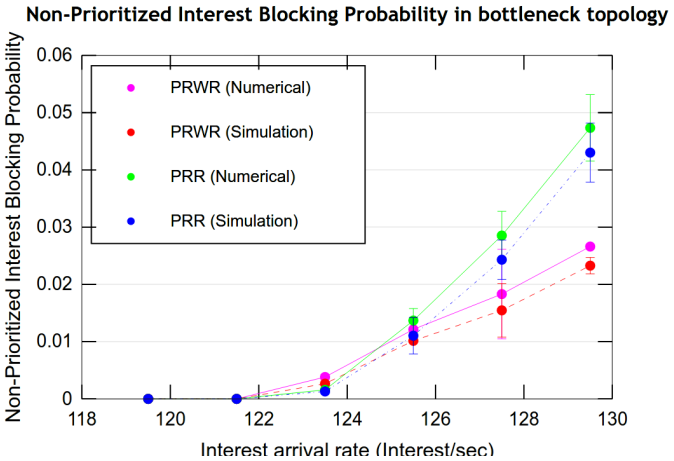


Figure 3.18: Prioritized Interest Blocking probability in Rocketfuel topology. The error bars in the plot indicate 95% confidence intervals. The simulation results are same both for 30% and 40% prioritized Interests.

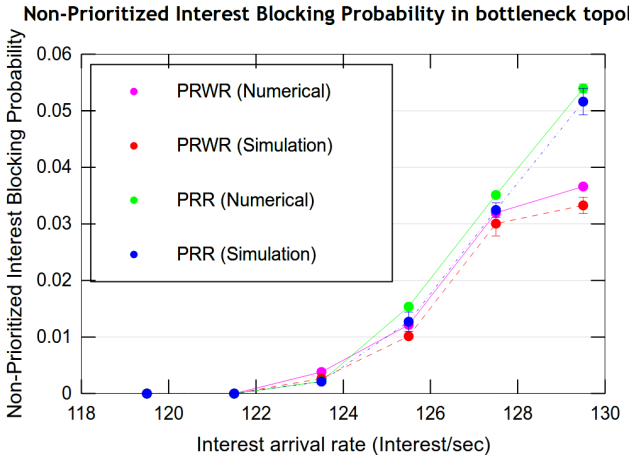
As shown in Figures 3.21, 3.22 and 3.24, the non-prioritized forced termination probability P_{nft} for both schemes grows as the number of prioritised Interests increases. This occurs because as the number of prioritised Interests increases, the number of removed non-prioritized entries decreases. However, despite increasing Prioritized Interests, P_{nft} for the PRR scheme does not rise due to reservation.

3.7 Summary

In this chapter, we have investigated both PRWR and PPR schemes which are aimed to provide QoS to the premium users. We use the Continuous Time Markov Chain (CTMC) model to present the analytical models of both schemes and compute the performance measures, namely non-prioritized Interest blocking probability and non-prioritized forced termination probability. From the simulation and numerical results, it is obvious that PRR scheme outperforms the PRWR scheme at the marginal cost of a non-prioritized blocking probability. In case of a large number of prioritized Interests, PRR scheme with an optimally allocated number of reserved PIT entries better. However, for a small number of prioritized Interests, PIT entry reservation is not necessary.



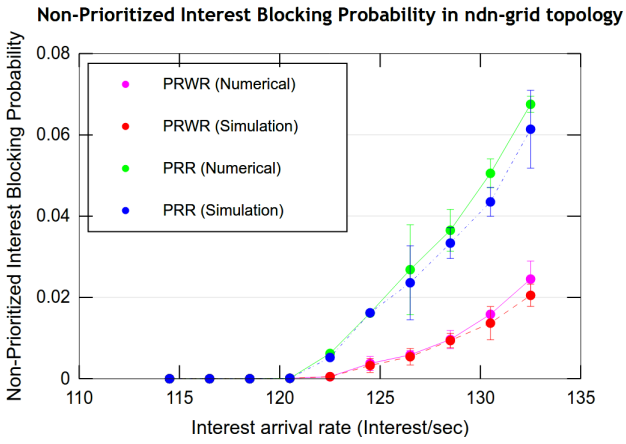
(a)



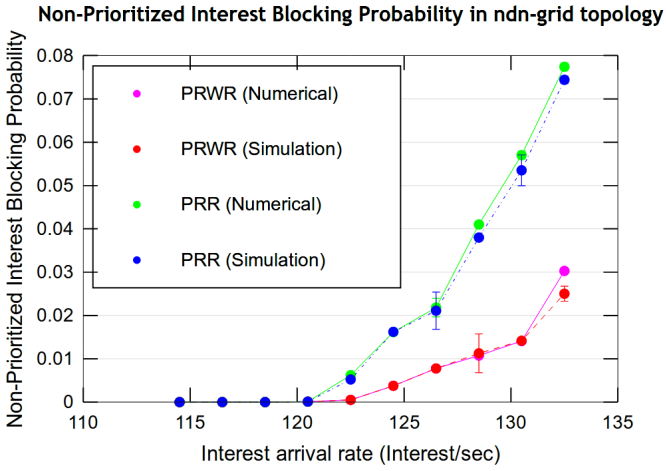
(b)

Figure 3.19: Non-Prioritized Interest Blocking probabilities for bottleneck topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals.

This chapter assumes that all consumers are genuine. However, in a real scenario, attackers can take advantage of the presence of PIT and make the PIT full. In the next chapter, we propose one attack model which is designed to degrade QoS of the legitimate consumers.

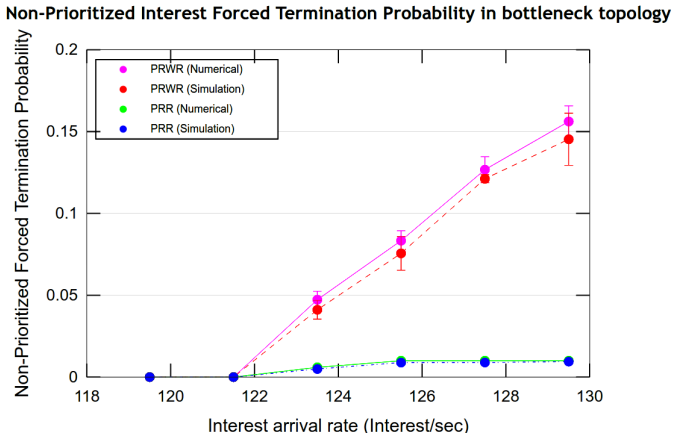


(a)

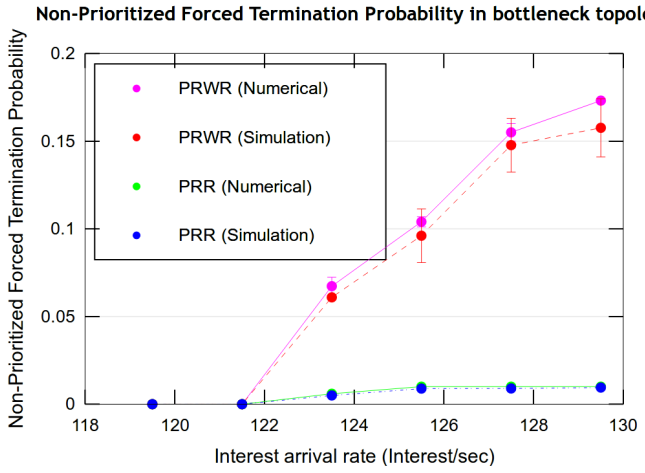


(b)

Figure 3.20: Non-Prioritized Interest Blocking probabilities for ndn-grid topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals.

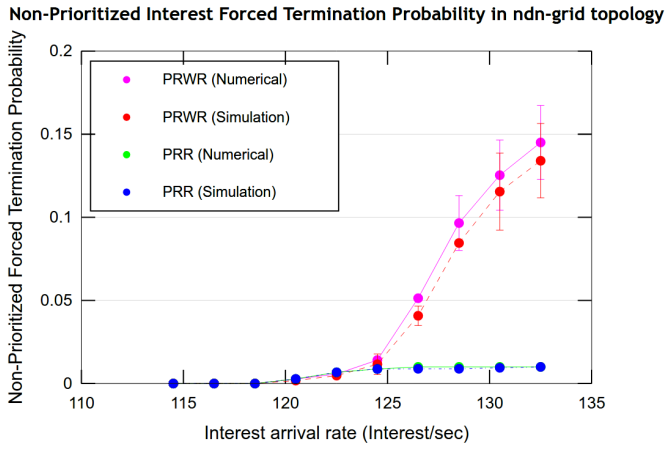


(a)

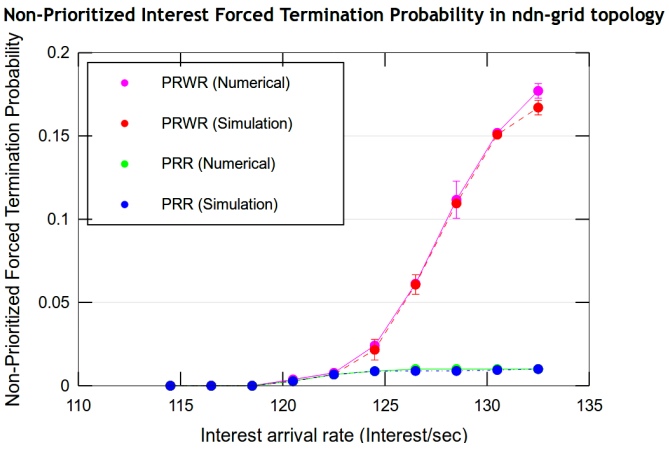


(b)

Figure 3.21: Non-Prioritized Forced Termination probabilities for bottleneck topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals

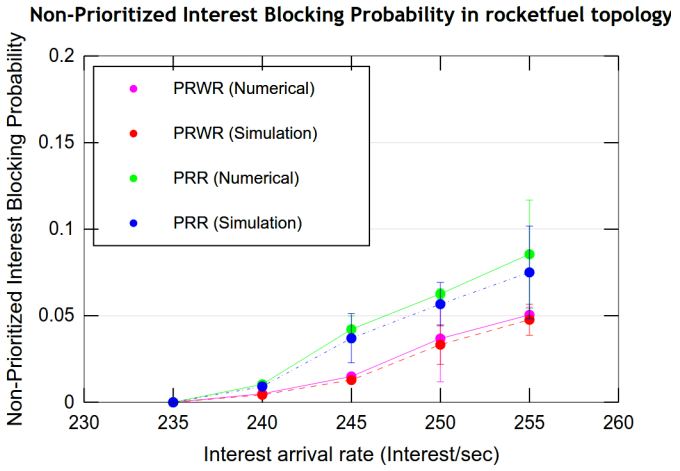


(a)

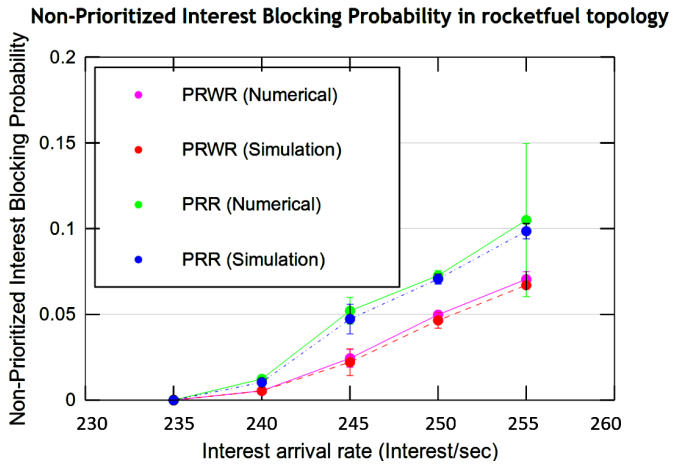


(b)

Figure 3.22: Non-Prioritized Forced Termination probabilities for ndn-grid topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals

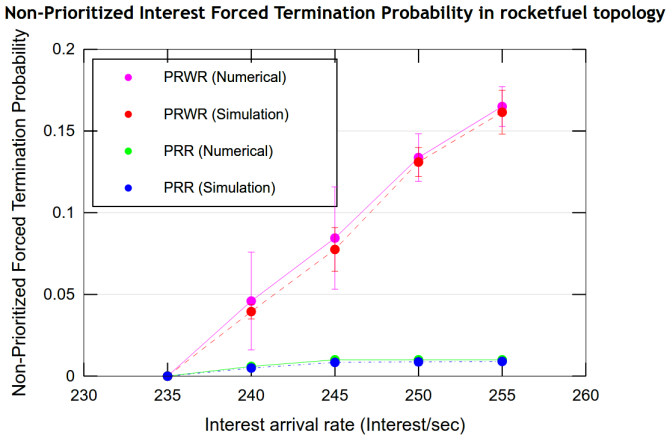


(a)

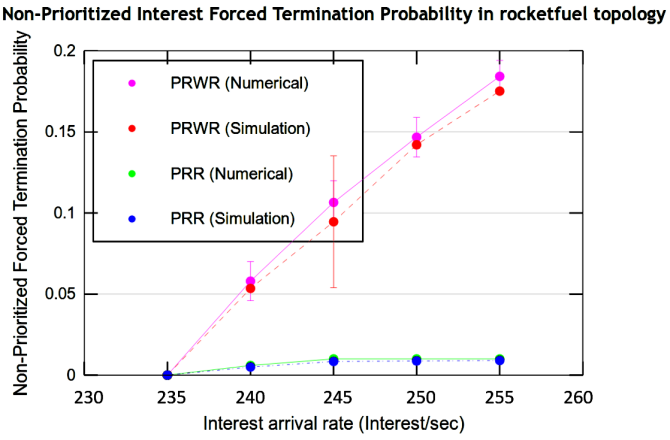


(b)

Figure 3.23: (a) Non-Prioritized Interest Blocking probability for rocketfuel topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals



(a)



(b)

Figure 3.24: (a) Non-Prioritized Interest Forced Termination probabilities for rocketfuel topology considering (a) 30 % prioritized Interests (b) 40 % prioritized Interests. The error bars in the plot indicate 95% confidence intervals

Chapter 4

SCAN: Smart Collaborative Attack in Named Data Networking

As we have already discussed in previous chapters, PIT stores the information of forwarded Interests so that later corresponding Data packets can be sent back to the requested consumer in the reverse path. We also discussed that PIT should have a fixed size. As we discussed in section 1.1, PIT facilitates various benefits to NDN such as anonymity, multicast, and multipath forwarding. Unfortunately, attackers can exploit its presence to launch an attack. Attackers can send a massive number of Interests with an aim to fill up the PIT and make it unavailable for legitimate consumers. Though in NDN, attackers can not directly target a particular consumer, they can focus and target on a router through which the targeted consumer's Interests are routed. In case the attacker successfully fills up the PIT of the router, incoming Interests get dropped, which results in a reduction in the Interest Satisfaction Rate (ISR) of the consumers. This attack falls within the Interest Flooding Attack (IFA) category, which has been thoroughly researched in [24, 25]. A recent study [26] revealed the existence of a collusion attack in which malicious consumers send requests (Interest packets) for contents from a malicious server. The attacker's goal is to increase content retrieval time. According to this study, if 20% of consumers are malicious and each sends 500 Interests/sec, content retrieval delay for legitimate users increases

by 20 times. Though they did not propose any solution, monitoring the PIT expiration rate can easily detect this attack. Moreover, due to the high-Interest sending rate of the attacker, a router can experience huge packet loss, which makes it easier to detect the potential attack. In such a scenario, existing countermeasures can identify the source of the attack or can decrease the attack's impact. We have also observed that most state-of-art countermeasures underestimate the attacker's potential. To the best of our knowledge, no one has explored that an attacker's attacking characteristics can change over time. In light of this, we propose a novel attacker model in which the attacker dynamically adjusts the Interest sending rate to succeed in the attack while remaining undetected. We have implemented our proposed attack model named SCAN in ndnSIM simulator [7], and the results validate the effectiveness of the attack.

The rest of this chapter is organized as follows: Section 4.1 describes the motivation behind our proposed attack model. Section 4.2 reviews some of the Interest Flooding Attack (IFA) works from the literature along with shortcomings of the existing Interest Flooding Attack (IFA) mitigation approaches. Section 4.3 provides the system model and assumptions for the proposed attack model. The details of the attack are described in section 4.4. The performance of the proposed attack is analysed using simulation results in section 4.5. Finally, section 4.6 presents a summary of the chapter.

4.1 Motivation

Even though there are many security solutions available in the current Internet, still we observe a large number of security attacks each year. However, unlike the current Internet, NDN has a "*Security by design*" goal from the very beginning. Therefore, it is essential to investigate NDN's robustness and resilience from all aspects, including existing and potential security threats. From the state-of-the-art literature, we observe that attackers' capabilities are undermined. For example, attackers can shift their characteristics over time. Furthermore, we discover many weaknesses in existing countermeasures (please refer to section 4.2 for more details). These factors influenced the development of our proposed attack model.

4.2 Related works

This section limits our discussion to Interest Flooding Attack (IFA). Please refer to [27] for a more comprehensive analysis of NDN security.

In IFA, attackers send out a large number of Interest packets to overload the routers' PIT or the targeted content producer. We can categorize IFA into three types: (a) Type I, attackers send Interest packets for *existing* content. However, because of in-network caching, the damage is limited. (b) In type II, attacker sends requests for *non-existent* content. These requests can easily be made by adding a random keyword to the valid Interest name prefix. (c) In type III, attackers send request for *dynamic* content. This attack is designed to overload a targeted producer. As an NDN principle, the producer should sign the published content, and signing requires significant computation overhead. So, too many requests for dynamic content overload the producer [24].

Afanasyev et al. [25] propose three IFA mitigation approaches: (i) Token bucket with per-interface fairness, (ii) Satisfaction-based Interest acceptance, and (iii) Satisfaction based Push-back (SBP). In comparison to the others, SBP performs better.

Token bucket with per-interface fairness method is an extension of the existing Token Bucket idea from TCP/IP architecture. To forward one Interest, a token is required. The number of tokens (Pending Interest Limit) for each outgoing interface is estimated from the bandwidth-delay product (BDP). However, in the presence of an attack, all tokens may be already used while forwarding Malicious Interest. To avoid this and make fairness among incoming interfaces, an individual queue is considered for each incoming interface of a router. Packets are forwarded from each queue in a round-robin fashion. Thus, Interests from one interface can not consume the entire outgoing link bandwidth. However, this scheme does not separate legitimate and malicious Interests. So, an attacker can use a large portion of the outgoing bandwidth.

In Satisfaction-based Interest acceptance, a router keeps an estimate of the Interest Satisfaction Rate (ISR) for each incoming interface. An incoming interest is probabilistically forwarded depending on ISR value. In this approach, since each router takes an independent decision of forwarding and rejecting an Interest, therefore, a large number of Data packets can not be forwarded back to the consumer due to the removal of PIT entries. So, with the increase in hop length, the consumer's ISR value decreases.

In Satisfaction based Pushback, a limit is maintained for each incoming interface depending

on ISR. The routers announce this limit to downstream neighbours so that they can restrict the malicious Interest from the attacker.

Poseidon [28] proposes a detection and mitigation approach for an attack where attackers send Interest packets for non-existent content at a high-Interest rate. As a result, routers' PITs become overburdened, causing legitimate Interest packets to be dropped. There are two aspects to Poseidon: detection and reaction. They have used two parameters for detection: *Interest satisfaction ratio* and *PIT usage per interface*. These parameters are monitored over a time period. The router enters mitigation mode when both parameters exceed their respective threshold limits. The router sends an alert message to its downstream neighbour routers. This aids in the early detection of an attack. This approach does not distinguish between legitimate and malicious consumers. Therefore, legitimate consumers sharing the same interface also get affected.

Nasserala et al. [26] propose a Collude Interest Flooding Attack (CIFA) with an aim to increase the content retrieval time for legitimate consumers. In this attack, attackers send Interests for contents stored by the malicious producer. Interests are sent at a high rate. Due to this, the malicious contents consume a large portion of caches of the intermediate routers. As a result, legitimate Interests need to be forwarded to the producer, which increases content retrieval delay. This work demonstrates the attack's feasibility. Due to the high-Interest rate, the malicious prefix's expiry rate becomes high. It leads to easy detection of the malicious prefix.

Salah et al. [29] present a detection and mitigation framework for a Collusive Interest Flooding Attack where malicious consumers issue Interests that can be served only by the malicious producer. In this framework, to monitor the ongoing traffic, a small group of routers are chosen to monitor the network traffic. These monitoring routers (MR) are chosen based on their proximity to the attack source and the number of routes that travel through it. Each MR calculates PIT utilization rate for (a) whole PIT (U_{Ri})and (b) per name prefix $/prefix_i$ [$U_{Ri}(/prefix_i)$]. If U_{Ri} is more than a threshold (α), then it considers that it is under attack. Later it checks $U_{Ri}(/prefix_i)$ for each name prefix $/prefix_i$. In case, if it finds any prefix's $U_{Ri}(/prefix_i)$ value more than a preassigned threshold (β), then it consider $/prefix_i$ as malicious. The drawback is that this detection fails in scenarios where attackers frequently change their prefixes.

Table 4.1: Summary of detection and mitigation of Interest Flooding Attack proposals

Approach	Detection	detection parameter	Mitigation	Shortcomings
Token bucket with per-interface fairness [25]	per router	none	forwards packets in a round-robin fashion from each incoming interface	do not separate legitimate and malicious interests
Satisfaction-based Interest acceptance [25]	per interface	ISR	forward Interests on the basis of ISR	consumer's ISR decreases due to each router's independent decision of accept/drop of an incoming Interest
Satisfaction based Pushback (SBP) [25]	per interface	ISR	throttle the traffic of the malicious Interests by setting a limit based on ISR	this approach fails in case of collaborative attack
Poseidon [28]	per interface	ISR and PIT usage	Limit PIT size and alarm is sent downstream	Legitimate consumer sharing the same interface with Malicious consumer gets affected
Salah et al.[29]	per name prefix	PIT utilization rate	reject Interests based on PIT utilization rate	does not work when attacker's frequently changes prefix

Shortcomings of existing IFA detection schemes

As we can observe in Table 4.1, the majority of attack detection schemes are based on two parameters: ISR and PIT usage. A router monitors these values over a period of time. However, these parameters are ineffective in some scenarios. For example, in a collaborative attack, where attackers request contents from malicious producers, a router can not differentiate between legitimate and potential malicious prefixes based on ISR. Therefore, ISR based attack detection is not feasible. Moreover, a router can experience a spike in PIT usage in the presence of bursty traffic. So, if we 'PIT usage' as an indicator for an attack, it would not be appropriate.

4.3 System Model and assumptions

4.3.1 System model

In our proposed attack model, we consider an NDN network consisting of six entities (refer to Figure 4.1).

- Legitimate Producer (LP): These producers are genuine and store valid contents.
- Legitimate Consumer (LC): These consumers are honest and request for valid contents stored by LP.
- Malicious Producer (MP): These evil producers store fake contents. However, the contents are signed with valid signatures so that intermediate routers can not detect the content by verifying the signature. Due to a valid signature, from a network perspective, the contents are valid.

- Malicious Consumer (MC): These dishonest consumers request for contents stored by MP.
- Monitoring Node (MN): These nodes are co-located with LC behind an edge router. They are compromised by an adversary. Monitoring nodes keep track of their own ISR. If the ISR falls below a certain threshold, an alarm is sent to MC. After reception of this alarm, MC decreases the Interest sending rate making it difficult to identify the ongoing attack.
- NDN router: Their function is to forward NDN packets.

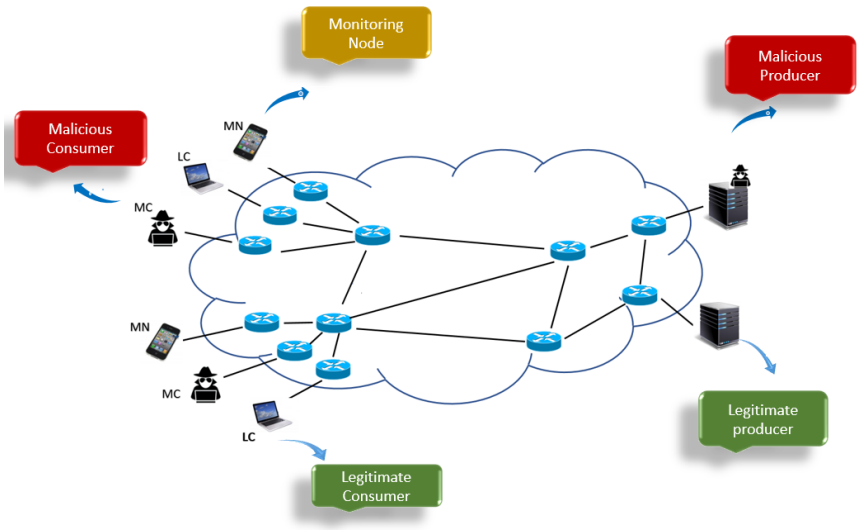


Figure 4.1: Different entities of SCAN attack

4.3.2 Assumptions

In our proposed attack model, we have considered the following assumptions:

- Attackers are aware of the topology. It is possible because certain autonomous systems provide information about their topology. Even if it isn't provided, we can still find the topology information [30, 31].
- Attackers have knowledge of the countermeasures installed in routers.
- It is possible for attackers to compromise only the end hosts (not routers).

4.4 Description of SCAN Attack

The high-level **goal** of SCAN attack is to degrade the QoS of the targeted legitimate customers (LC). Unlike IP, we can't directly target a particular consumer in NDN due to the absence of an IP address. Therefore, the attacker targets an intermediary router through which LC's Interest packets transit. The attacker's goal is to exploit the PIT of the targeted router and fill it up so that incoming Interests get dropped. To understand this better, we consider a toy example in Figure 4.2, where LC is a legitimate consumer while MC is a malicious consumer. LC sends Interests for contents that are stored in LP. The Interest traverses the path $LC \rightarrow R2 \rightarrow R4 \rightarrow R5 \rightarrow R7 \rightarrow$ LP. The satisfying Data packet corresponding to the Interest returns back to LC following the reverse path $LP \rightarrow R7 \rightarrow R5 \rightarrow R4 \rightarrow R2 \rightarrow LC$. Now, MC targets to fill up the PIT of the router $R5$. So, it sends Interests towards MP through router $R5$. In case it is successful in filling up the PIT, the incoming Interest packets from LC get dropped. Thus, LC's ISR is reduced causing degradation of QoS.

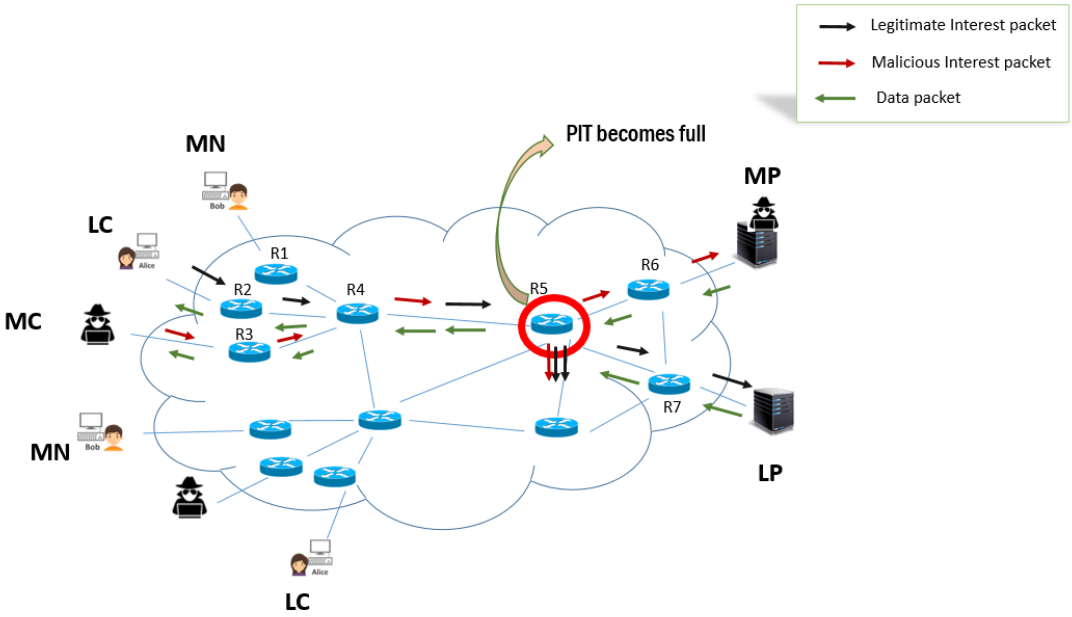


Figure 4.2: Toy example of SCAN attack.

Now, the question arises: what type of Interest does the attacker send? They have two options: they can send genuine requests for valid content or fraudulent requests for invalid

content. In case of valid contents, the intermediate routers cache these contents for future use, which reduce the impact of the attack [24]. On the other hand, if they send Interests for invalid content (which can not be served by any node or producer), the intermediate router has to wait until PIT expires, which causes PIT to fill up quickly. The name of the fraudulent request can be easily made by adding arbitrary keywords to a genuine Interest name prefix. However, the existing mitigation approaches [28] showed that this kind of attack might be detected and handled by looking at parameters like ISR and PIT usage. On the other hand, if a group of malicious producers are working together with malicious consumers to launch an attack (collaborative attack), in such a scenario, the parameters like ISR are not valid for detection. Therefore, in our proposed attack model, we consider the case of collaborative attack so that a router can not distinguish between malicious and legitimate prefixes based on ISR.

Even though the idea looks simple, it is difficult for attackers to successfully carry out the attack. The attacker faces the following **challenges**:

- According to the principles of NDN, an NDN packet should always be signed by the producer. Since the signing process is costly and time-consuming; so a producer can not always generate signed packets dynamically. Furthermore, a malicious producer can not have infinite resources. As a result, an attacker requires an approximate estimation of the number of contents that must be stored in the malicious producer.
- Another challenge is that if attackers send Interests with high frequency, a router can experience a huge packet drop, which may help easy detection. The question arises: at what rate should a malicious consumer send Interest?

To overcome these challenges, attackers have two **solutions**:

- Attackers can calculate the number of contents that are to be stored in malicious producer beforehand.
- Attackers need to set the Interest rate dynamically to avoid the attack detection.

The proposed attack model is designed to incorporate the above solutions. Figure 4.3 illustrates our SCAN attack model. It has two phases.

- **Pre-Attack Phase**: In this phase, the attacker calculates the total number of unique contents that need to be stored in the malicious producer (N_{mp}). For calculation of

N_{mp}, we have to estimate the Minimum Retransmission Wait Time (ΔT) and Minimum frequency (m_freq). The reason behind the estimation of ΔT and m_freq are discussed later.

- **Attack Phase**: In this phase, to evade detection and match with dynamic traffic, the attackers dynamically change the Interest rate.

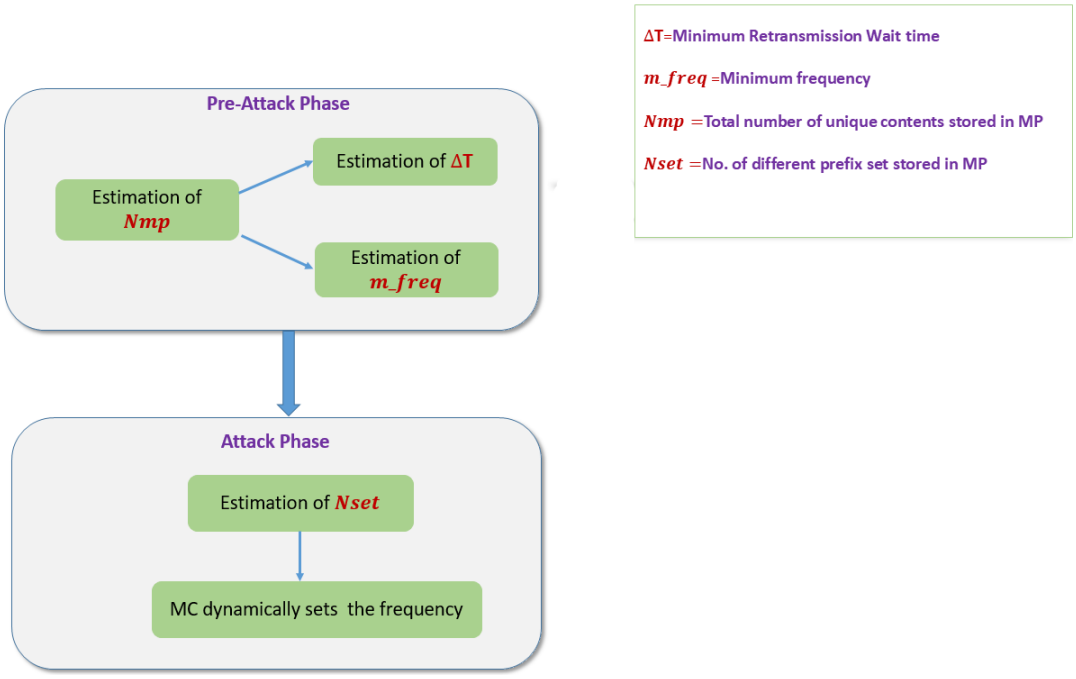


Figure 4.3: Different phases of SCAN attack

The details of each phase are explained below.

4.4.1 Pre-Attack Phase

A MP can have a fixed number of pre-signed contents. So, in this phase, the attacker tries to estimate the number of contents to be stored in MP. The attacker's target is to fill up the specified router's PIT, which is directly/indirectly connected to LP). For that, Interest packets from MC must not be served by any router in between MC and the targeted router. To satisfy this condition, MC must request contents with a unique name. Otherwise, the attack may not be successful. One technique to get around this is to re-transmit packets that have already been

sent at some time. However, the time interval between retransmissions should be such that the matching Data packet is removed from all routers' caches between MC and the targeted router. For convenience, we term this time interval as "*Minimum Retransmission Wait Time (ΔT)*".

Once the attacker has the value of ΔT, it can calculate the number of contents to be stored in MP (N_{mp}) as showed in Equation (4.1).

$$N_{mp} = \sum_{i=1}^{n} \Delta T_i * m\_freq_i \tag{4.1}$$

where ΔT_i and m_freq_i are *Minimum Retransmission Wait Time* and *Minimum Interest frequency* for i^{th} MC. n is the total number of MC present in the topology. The reason behind this equation is: an attacker cannot send the same packet for ΔT_i time period and it sends m_freq_i number of Interest packets per second. Therefore, N_{mp} is the product of ΔT_i and m_freq_i.

Now, the question comes to our mind is: how we can calculate ΔT_i and m_freq_i? We describe the calculation process next. Since the same estimation process is followed by all attackers; we omit the subscript i for simplicity.

(i) Estimation of Minimum Retransmission Wait Time

The estimation of *Minimum Retransmission Wait Time* (ΔT_i) is described in Algorithm 6 and illustrated in Figure 4.4 for clear understanding. To estimate ΔT_i, MC sends an Interest packet to MP. After receiving the Data packet, it calculates the RTT. RTT for a consumer is the time interval between sending an Interest and receiving the corresponding Data packet (refer to Figure 4.5).

We can see in Algorithm 6, the function $SendNewPacketGetRTT()$ calculates the RTT value R_0. The attacker ensures that the Interest which is unique so that it does not come from a network cache. After that, MC waits for a time interval (we consider the initial value as 100 msec) and then retransmits the same Interest. After receiving the Data packet, it calculates the RTT (R). In Algorithm 6, $SendPrevPacketGetRTT()$ calculates R. Now, MC compares both values: R_0 and R. If R is smaller than R_0, that means the content has come from an Intermediate router's cache. Therefore, MC increase the wait time (2*100 msec). After waiting for 200 msec, it again retransmits the same Interest packet. Again it calculate RTT value (R). In case, the $R < R_0$, MC increases the waiting time period to 2*200 msec and it continues. Suppose, after waiting for 400 msec, MC sends the same Interest and receives a Data packet.

This time RTT is not less than R_0. So, now MC probes for an optimal time interval between 400 and 200 msec using binary search.

Algorithm 6: Algorithm to find ΔT

Input: ΔT /*initial minimum Retransmission wait time*/, ae=0.001 /*Allowable error*/

Output: optimal value of ΔT

```
1  flag=0
2  R_0 ← SendNewPacketGetRTT()   // R_0 holds the RTT value for the first transmission
   of Interest packet
3  Wait for ΔT
4  R ← SendPrevPacketGetRTT()    // R holds the RTT value for other transmission of
   Interest packet
5  while R < R_0 do
6      ΔT' ← ΔT
7      ΔT ← 2 * ΔT
8      R ← SendPrevPacketGetRTT()
9      wait for ΔT
10 while (ΔT' < ΔT)AND(flag == 0) do do
11     ΔT'' ← (ΔT'' + ΔT'/2) wait for ΔT''
12     R ← SendPrevPacketGetRTT()
13     if R < R_0 then
14         ΔT' ← ΔT''
15     else if ΔT' = ΔT + ae then
16         flag = 1
17         return ΔT                          // return the optimal value of ΔT
18     else
19         ΔT ← ΔT''
```

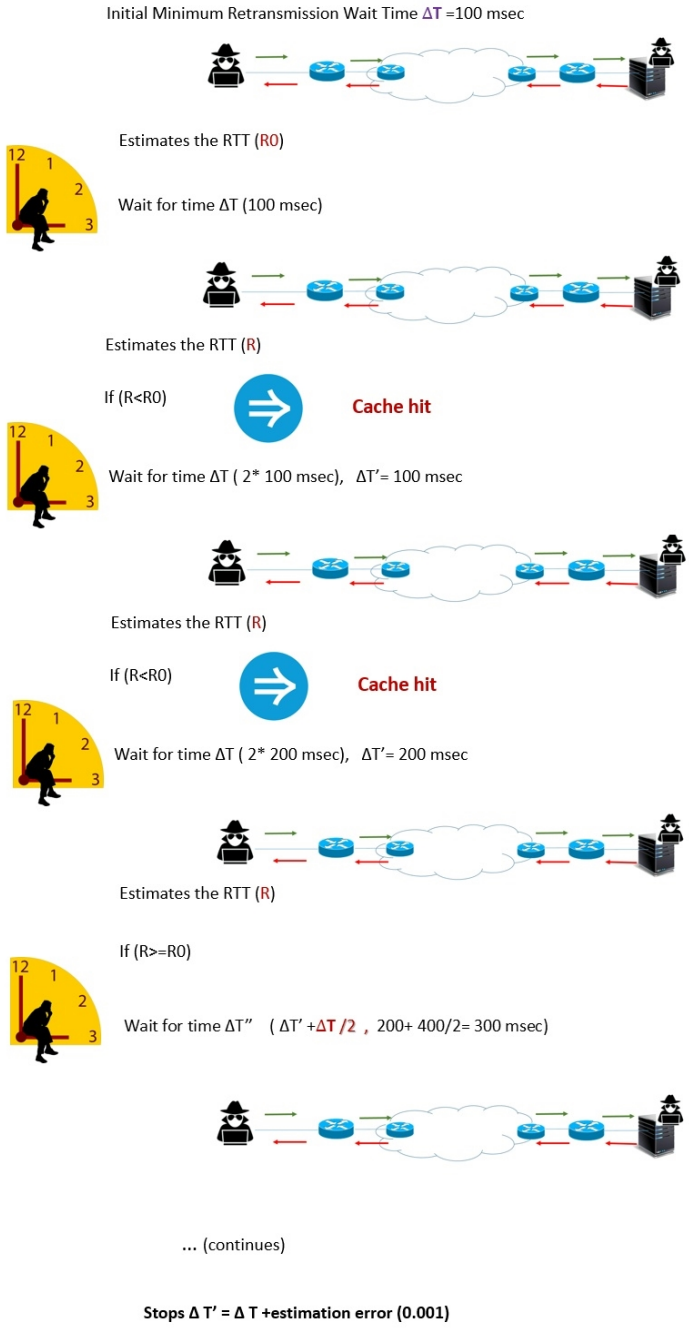


Figure 4.4: Demonstration of estimation of ΔT

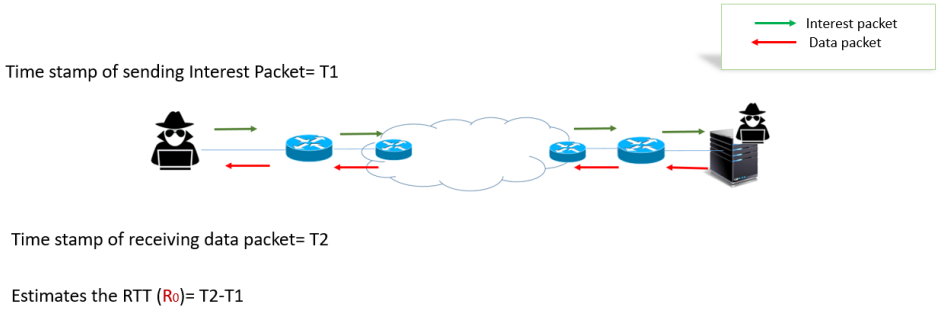


Figure 4.5: Demonstration of estimation of Round Trip Time (RTT) by MC

(iii) Estimation of Minimum Interest Frequency (m_freq)

Algorithm 7: Calculation of ISR by MN

Input: $m\_Interest$=0 /* total number of forwarded Interest*/, m_Data=0 /* total number of incoming Data packets */, $avg\_Interest$=0 /* average value of $m\_Interest$ */, avg_Data=0 /* average value of m_Data */, α =0.07 /* constant weighting factor */, isr=0.0 /* Interest Satisfaction Rate */

Output: isr /* value of ISR*/

1 function *OutInterest (Interest i)*

2 $m\_Interest = m\_Interest + 1$

3 function *InData (Data d)*

4 $m\_Data = m\_Data + 1$

5 function *Isr_Calculation()* `// this function is calculated after each time interval T`

6 $avg\_Interest = \alpha * avg\_Interest + (1 - \alpha) * avg\_Interest$

7 $avg\_Data = \alpha * avg\_Data + (1 - \alpha) * avg\_Data$

8 $isr = avg\_Data/avg\_Interest$

9 **if** $isr > threshold$ **then**

10 send an alert to the malicious consumer

11 **else**

12 do nothing

13 $avg\_Interest$ =0

14 avg_Data=0

Algorithm 8: Estimation of Minimum Frequency by each MC

Input: $m\_Interval$=10 sec /* interval at which the attacker changes the Interest frequency*/, m_freq=0 /* minimum Interest frequency, initial value set to 10 Interest/sec*/, *firstTime*=true, *flag*==0

Output: m_freq /* minimum Interest frequency*/

```
1  function StartApplication()
2  ScheduleNextPacket()
3  Simulator::schedule (Seconds( m_Interval& ScheduleChange, this)
4  function ScheduleChange() .      // After receiving an alert from neighboring Monitoring
   Node (sets flag value to 1)
5  if flag==0 then
6      m_freq = m_freq * 2
7  else
8      return m_freq                                  // returns the minimum frequency
9  Simulator::schedule (Seconds( m_Interval& ScheduleChange, this)
10 function ScheduleNextPacket()
11 if firstTime==true then
12     Simulator::schedule (Seconds( 0.0 & SendPacket, this)
13     firstTime=false
14 else
15     Simulator::schedule (Seconds( 1.0/ m_freq& SendPacket, this)
16 function SendPacket()
17 Create a new Interest and send it
18 ScheduleNextPacket()
```

For estimation of m_freq, an attacker MC compromises a consumer node near the targeted legitimate consumer behind the same edge router. This compromised node behaves like a legitimate consumer and sends requests for contents from LP. These nodes are known as *Monitoring Node (MN)*. The additional job of MN is to calculate the ISR per time interval and send an alarm to MC if ISR value is lower than a predefined threshold. Function $Isr\_Calculation()$ of Algorithm 7 shows the estimation of ISR by MN. ISR value is calculated

from the average value of total Data packet received avg_Data and total Interest packets forwarded $avg\_Interest()$. For calculation of the average, we have used Exponentially Weighted Moving Average (EWMA) formula (refer to line numbers 6 and 7 of Algorithm 7). α is the balancing factor.

Initially, the attacker starts sending Interest at a low frequency (suppose 10 interests/sec). After a particular time interval (suppose 10 sec), the attacker increases the frequency two times its previous value. The frequency is increased until the attacker receives an alert from MN. Function *ScheduleChange()* of Algorithm 8 shows the calculation of minimum Interest frequency (m_freq).

So, in this Pre-attack phase, after calculation of *Minimum Retransmission Wait Time* (ΔT) and *Minimum Interest frequency* (m_freq), we calculate total number of unique contents to be stored in MP (N_{mp}) as shown in Equation (4.1). A question may arise: since the network is dynamic and the parameters chosen at the beginning of the attack may not be optimal in later parts of the attack. Please note that the attack will be launched for a short duration. The attacker will become silent after the attack period. In the worst case, if the attackers fail to successfully launch the attack, they have to re-estimate the parameters.

4.4.2 Attack Phase

In this phase, the attacker uses the information obtained during the Pre-Attack phase. Though *Minimum Interest frequency* (m_freq) information is available, to avoid attack detection and to cope with traffic fluctuations, the attacker sets the frequency dynamically.

In addition to that, the attacker changes the name prefix periodically to raise the attack detection bar. As a result, the producer must keep various contents under several prefix names.

The number of different prefix sets can be calculated as follows:

$$N_{set} = N_{mp}/(n * m\_Interval) \tag{4.2}$$

where N_{set} is the number of different prefix sets, N_{mp} is the total number of unique contents estimated during the pre-attack phase, n is the number of malicious consumers, and $m\_Interval$

is the time interval after the attacker changes the name prefix.

Algorithm 9: Dynamic setting of Interest Frequency by each MC

Input: $current\_Ifreq = m\_freq/2$ /* current Interest frequency */, $m\_Interval$=10sec /* interval at which the attacker changes prefix name of the Interest packet and changes Interest Frequency*/, *isFirst*=true

```
1  function Start()
2  ScheduleNextPacket1()
3  Simulator::schedule (Seconds( m_Interval& PrefixChange, this)

4  function PrefixChange
5  if flag==0 then
6      curr_Ifreq = curr_Ifreq
7      curr_Ifreq = (prev_Ifreq + m_freq)/2
8  else
9      curr_Ifreq = (prev_Ifreq + curr_freq)/2
10 Simulator::schedule (Seconds( m_Interval& PrefixChange, this)

11 function ScheduleNextPacket1()
12 if isFirst==true then
13     Simulator::schedule (Seconds( 0.0 & SendPacket, this)
14     isFirst=false
15 else
16     Simulator::schedule (Seconds( 1.0/ curr_Ifreq& SendPacket1, this)

17 function SendPacket1()
18 Create a new Interest and send it
19 ScheduleNextPacket1()
```

Algorithm 9 illustrates how an attacker changes the Interest frequency after some time interval depending on the alarm sent by the Monitoring Node (MN). To understand it better, we can consider one example. Suppose, in pre-attack phase, attacker get the value of m_freq= 800 Interest/sec. So, $current\_Ifreq$=800/2=400 Interest/sec. So, initially, the attacker starts sending Interest at 400 Interest/sec. After each time interval $m\_Interval$, it checks whether it

receives an alert from MN. In case there is no alert, the attacker increases the Interest frequency. Otherwise, the attacker decreases the frequency. For increase and decrease, we use binary search (refer to line number 5-9 in Algorithm 9).

Please note that attacks are usually mounted for a relatively short duration. So during the attack, network dynamics may remain stable. Again, if attacker parameter estimations are not correct, attack launching may not be successful. As a result, every time, estimation of the parameters is necessary for a fresh attack to be successful.

4.5 Performance Analysis

The SCAN attack model is implemented in ndnSIM simulator [7] (version 2.6). The performance of the proposed attack is analyzed for the effectiveness of the attack. The configuration of the system that we used for the experiment: Ubuntu 16.04, memory: 16GB, processor: corei7-6700 CpU@3.40GHz. Further, we implement three state-of-the-art IFA mitigation mechanisms [25, 28, 29] in ndnSIM and evaluate them against our attack model to verify the applicability of the attack in NDN.

4.5.1 Evaluation Setup

Figure 4.6 depicts a realistic AT & T network topology that we used for our experiment. In the Figure, Cx, Ax, Mx, Rx, LPx, and MPx denote the x-th consumer (legitimate), adversary (malicious consumer), monitoring node, Legitimate Producer and Malicious Producer, respectively. The lines show how the nodes are linked together. In our experiment setup, we consider 15 Legitimate consumers (LC), 3 Malicious consumers (MC), 2 Monitoring nodes (MN), one Legitimate producer (LP) and one malicious producer (MP). Each LC issues Interest with Zipf-Mandelbrot distribution. We keep Interest Lifetime as its default value (4 sec). Each Data packet (1054 bytes) is assumed to be a single content item. To avoid cache hit inside the MC itself, we disabled CS. Other nodes have CS size=50 content items. We use Least Recently Used (LRU) content replacement policy. LCs send Interests at t=1 sec and stops at t=540 sec and MCs starts at t=60 sec and stops at t=540 sec. LC sends Interests at 50 Interest/sec and follows a Zipf Mandelbrot distribution with α=0.7.

The simulation parameters are listed in Table 4.2. In the previous chapter, we have already discussed the importance of limiting the PIT size. We can calculate the PIT size of each router (PIT_{size}) as follows:

$$PIT_{size} = N_{oi}.\frac{(BW[bytes/sec] * Delay[sec])}{Size_{Dpacket}[bytes]} \tag{4.3}$$

where N_{oi} is the number of outgoing links, BW is the total bandwidth of the outgoing links, $Delay$ is the average time required to satisfy an Interest, $Size_{Dpacket}$ average size of the received Data packets. The accurate estimation of these values is not possible. It is not necessary as well. We can take average values based on the network traffic. The fluctuation of these values can be smoothened out by network buffers.

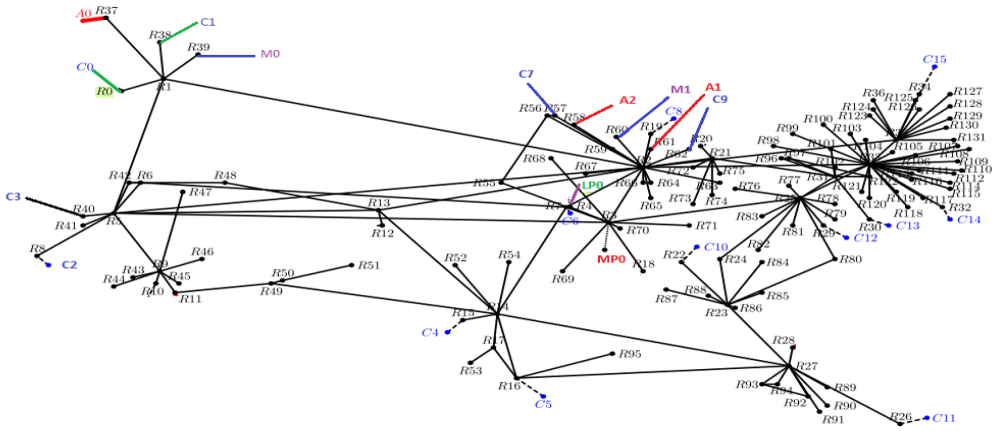


Figure 4.6: AT & T topology: Legitimate Consumer (Cx), Adversary or Malicious Consumer (Ax), Monitoring node (Mx), Router (Rx), Legitimate Producer(LPx) and Malicious Producer (MPx)

Table 4.2: Simulation parameters in ndnSIM simulator

Parameter	*Value*
Forwarding Strategy	Best-Route Strategy
CS Replacement Policy	LRU
Content Store size	50
Interest lifetime	4000 ms
Frequency of Legitimate consumer	50 Interest/sec
Interest Packet Size	215 Byte
Data Size	1054 Byte
Zipf Factor α	0.7
Simulation Time	540 sec

4.5.2 Evaluation metrics

The effectiveness of the proposed SCAN attack model is quantified using the following metrics:

- *Average Interest satisfaction ratio (ISR) of legitimate consumers:* It is the ratio of the total number of received Data packets to the total number of forwarded Interest packets. It denotes the quality of service perceived by legitimate consumers.
- *PIT usage:* The maximum value of the PIT size during the simulation.
- *Average content Retrieval Time:* The average time interval between issuing an Interest and receiving the corresponding Data chunk back.
- *Average Goodput:* Average number of Data packets received by Legitimate consumers per unit time.

4.5.3 Results

(A) *Attack Validation*

In **Pre-attack Phase** we implement Algorithm 6 to measure Minimum Retransmission Wait Time (ΔT). From the simulation experiment, we have found the value of ΔT for C0, C1 and C2 are 38 sec, 15 sec, and 14 sec, respectively. We also get the value of Minimum Interest Frequency

m_freq for attackers A0, A1 and A2 are 640 and 320, 320 Interest/sec, respectively. So, we calculate the number of Data packets (N_{mp}) that to be stored in Malicious producer(MP) using Equation (4.1) [38*640+ 15*320+ 14*320)=33,600].

In **Attack Phase** phase, we make use of the data/information gathered during the *Pre-attack phase*. To validate the effectiveness of the attack, we calculate the performance metrics (discussed in 4.5.2) for both the baseline scenario (no attack) and the attack scenario.

Figure 4.7 shows the Average Interest Satisfaction Rate (ISR) of Legitimate consumers. In comparison to baseline behaviour, we observe a decrease in satisfaction rate of up to 21% (highest) during the SCAN attack. It is important to observe that the proposed attack model does not result in a major decline in customer satisfaction. Hence the attack goes unnoticed. We used Algorithm 9 to dynamically set the Interest Frequency for attackers to make the attack successful.

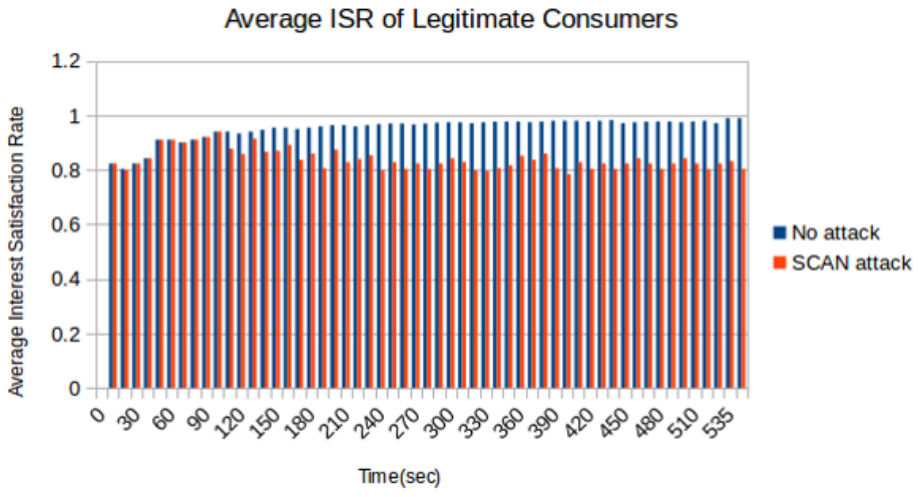


Figure 4.7: Average Interest Satisfaction Rate (ISR) of Legitimate consumers

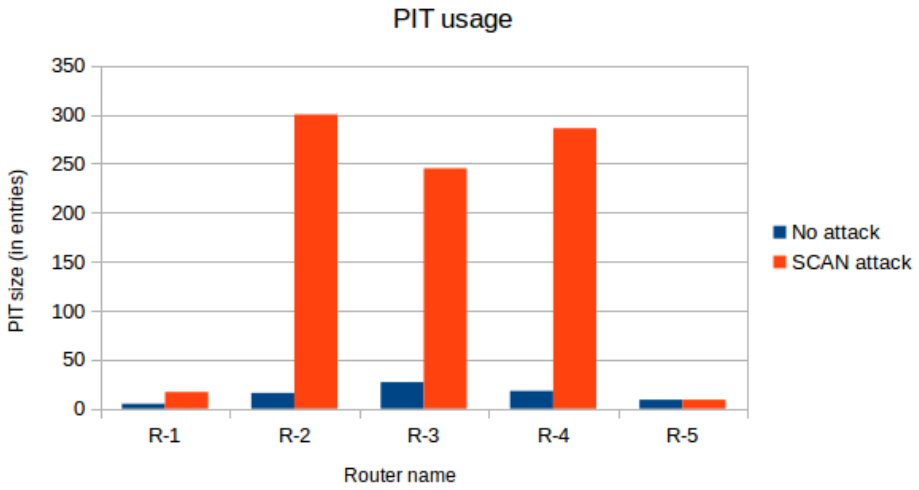


Figure 4.8: PIT Usage of different Routers

The PIT usage of different routers is depicted in Figure 4.8. Since we set PIT size according to Equation (4.3), the routers with a large number of outgoing interfaces and a long delay have a larger PIT size. During the attack, we observe a considerable rise in PIT usage in routers R2, R3, and R4 compared to the baseline scenario.

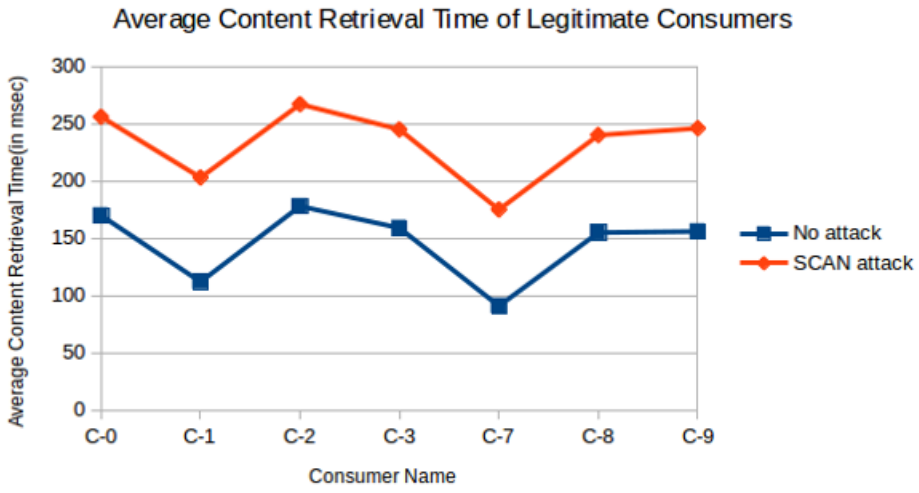


Figure 4.9: Average content Retrieval Time of all Legitimate Consumers

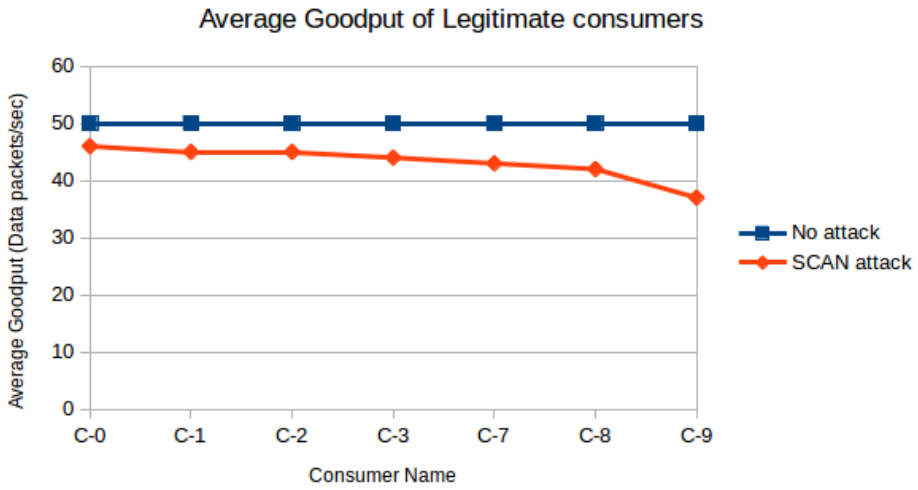


Figure 4.10: Average Goodput of all Legitimate Consumers

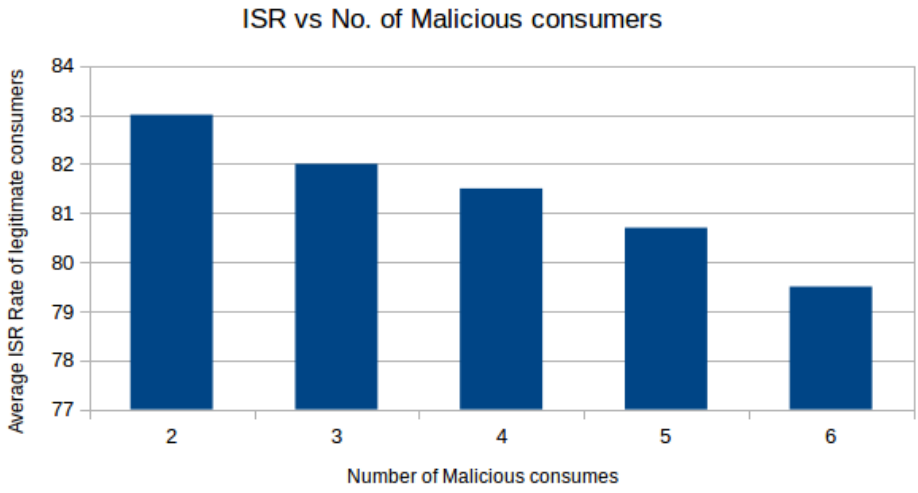


Figure 4.11: Effect of number of malicious consumers on ISR of legitimate consumers

Figure 4.11 illustrates the Interest satisfaction Rate with the increase in malicious consumers. Despite the number of malicious consumers has increased, the Interest Satisfaction Rate has remained relatively constant. It occurs because of the dynamic setting of the Interest Arrival rate for malicious consumers.

(B) Validating against existing IFA mitigation mechanisms

We have considered three state-of-the-art IFA mitigation mechanisms to validate our proposed SCAN attack as follows:

(i) *SBP* [25]

(ii) *Poseidon* [28]

(iii) *Salah et al.* [29]

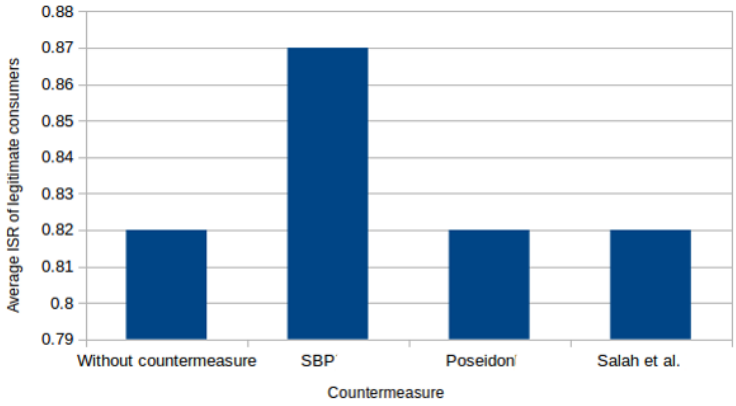


Figure 4.12: SCAN attack against existing countermeasures (ISR)

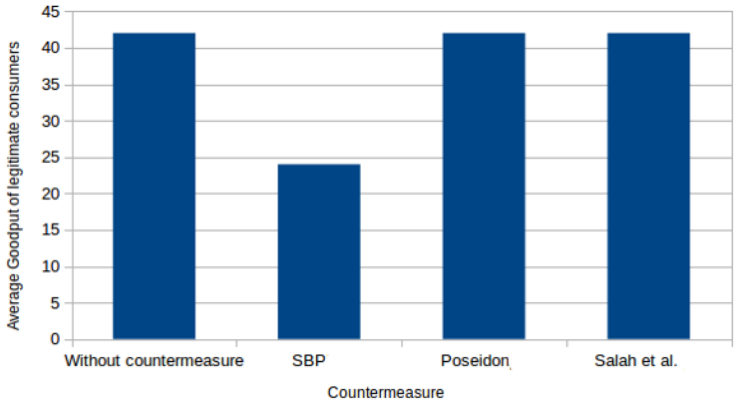


Figure 4.13: SCAN attack against existing countermeasures (Goodput)

We observe the simulation results in Figures 4.12 and 4.13. We notice that SBP has highest ISR among all the schemes. It is because SBP restricts the number of transmitted Interests

based on the interface's ISR. SCAN is a collaborative attack; therefore, even if we apply SBP, routers are unable to distinguish malicious prefixes based on ISR. Due to the restriction of forwarded Interests, Goodput decreases in SBP.

In Poseidon [28], detection of ongoing attack depends on ISR threshold which is set as 0.33. In SCAN attack, attacker's aim is to keep the ISR of the targeted consumers below a predefined threshold (0.8). So, clearly Poseidon fails to detect SCAN attack. Moreover, Poseidon is designed to mitigate IFA attack type (ii) where malicious customers request contents that do not exist. Since Data packets are not returned for those malicious prefixes, therefore, ISR for malicious prefixes is minimal. So, it is easy for the router to distinguish between good and bad Interests.

The mitigation approach suggested by Salah et al. [29] is also ineffective in detecting SCAN attacks. After a fixed observation period of 10 seconds, malicious customers modify the Interest prefix in our proposed SCAN attack. As a result, if monitoring routers identify a prefix (e.g., *'/evil1'*) as the malicious prefix, it is impossible to block malicious customers' Interests after the defined observation period.

In the conclusion of this discussion, we can assert that present countermeasures are ineffective against SCAN attack.

4.6 Summary

This chapter proposes a novel smart collaborative attack model for NDN with an aim to reduce the QoS of targeted consumers. The uniqueness of this attack is that there is no sudden decrease in Interest satisfaction Rate, due to which it is difficult for legitimate consumers/routers to detect it. Furthermore, distinguishing between legitimate and malicious traffic is not easy. As a result, traditional IFA solutions in the literature are ineffective in dealing with this attack. From a top-level view, it appears that congestion control schemes can simply fix the attack. However, in that case, legitimate traffic is also impacted. Finally, we firmly believe that developing countermeasures for such a sophisticated form of attack is essential for the real and successful deployment of NDN.

In this chapter and in chapter 3, we have observed that the presence of bursty traffic and attack leads to network congestion. In the next chapter, we propose one congestion control scheme, which is designed from PIT perspective.

Chapter 5

LPECN: Leveraging PIT placement and Explicit marking for Congestion control in NDN

In Chapter 3, we have already discussed the need to fix the PIT size. Overallocation of memory to PIT may harm network performance, such as a reduction of throughput and an increase in content retrieval delay. From our earlier works, we have observed that the network becomes congested in the presence of bursty traffic or Distributed Denial of Service (DDoS) attacks. Although a vast literature is already available in TCP/IP, we can not apply them directly to NDN for the following reasons: (i) in TCP/IP, communication happens between two nodes. However, in NDN, a consumer may receive Data packets from multiple sources because of in-network caching. Due to the network's dynamic nature (frequent eviction from the cache, huge traffic), no one can guarantee from which node the next Data packet may arrive. So, we can not use the existing TCP/IP's Retransmission Time Out (RTO) estimation scheme directly to NDN. (ii) Presence of in-network caches generalize the out-of-order delivery of Data packets. As a result, using three duplicate acknowledgements to detect congestion is irrelevant. Keeping these limitations in mind, Researchers have proposed several congestion control schemes in NDN. However, the majority of these works overlook PIT's existence. Since we already discuss the PIT's role in network performance in our previous chapters, so, in this chapter, we intend to

investigate congestion control from a PIT perspective. We propose a congestion control scheme named LPECN, which uses PIT per outgoing face placement to limit Interests based on available bandwidth. Please keep in mind that LPECN is designed to perform best-route forwarding; if more flexible forwarding is required, we have to opt for another design. Moreover, we have observed that none of the existing congestion control schemes considered the presence of non-responsive consumers. These consumers may not reduce the Interest sending rate even after receiving a congestion signal. Our proposed congestion control scheme also considers this fact.

The contributions of this chapter are summarized as follows:

- We deploy PIT per outgoing face placement to regulate Interests based on the capacity of the outgoing link.
- We use CUBIC [32] congestion control scheme at consumers for scalability and stability in long and fast distance networks.
- To manage excessive delay at network buffers, we implement CODEL Active Queue Management (AQM) scheme [33] at routers.
- For detection and limitation of Interests from unresponsive consumers, we use NACK and explicit marking. This is feasible because of NDN's architectural design features such as *symmetric forwarding* and *stateful forwarding plane*.
- Extensive simulation findings show that even in the presence of non-responsive consumers, the proposed system effectively controls congestion.

The rest of this chapter is organized as follows: Section 5.1 presents the motivation behind our work. After describing related work in Section 5.2, we explain the rationale behind considering PIT per outgoing face placement in section 5.3. The proposed scheme is discussed in section 5.4. Section 5.5 includes the performance analysis. Section 5.6 provides summary of the chapter.

5.1 Motivation

From the literature survey, we have observed that most congestion control approaches do not pay much attention to the presence of PIT. Moreover, they make an assumption that all consumers respond to congestion signals, which is not always true. There may be non-responsive consumers

who may send traffic bursts, occupying an ample space in the PIT table. It may trigger packet dropping in case PIT attains its full limit. Due to packet dropping, consumers experience packet timeout. The responsive consumers, in response to timeout, reduce the congestion window. Due to this, the Interest sending rate of the consumer decreases, leading to a heavy reduction in throughput. This chapter addresses this problem and leverages PIT per outgoing face placement and NDN's architectural features: *Stateful forwarding plane* and *symmetric forwarding.*

5.2 Related Works

In Content-Centric TCP (CCTCP) [21], consumers send anticipated Interests along with an actual Interest and forward them towards possible producers. It does this to identify the locations of the contents beforehand. After that, it maintains Retransmission Time Out (RTO) for each content source.

Wang et al. [34] propose an Interest shaper to regulate the returning Data rate and avoid congestion. The proposed design leverages NDN design features such as *receiver-driven communication* and *symmetric bidirectional forwarding.* This Interest shaper also considers the interdependence between Interests and returning Data, ensuring proportional fairness between two-way traffic. They have taken a mathematical approach and formulated an optimization problem to estimate the best optimal shaping rate. They have also proposed a practical Interest shaping algorithm to increase link utilization and reduce congestive loss.

Mahdian et al. [35] present a multipath aware rate-based congestion control scheme (MIRCC). The proposed design has been inspired by a rate-based congestion control scheme named RCP [36] in TCP/IP. The key idea behind RCP is that end-points are provided with a sending rate which is increased or decreased depending on whether or not there is congestion. Each node calculates the rate $R(t)$ for each outgoing link. On receiving Interest, the producer returns the satisfying Data packets and a maximum rate value (r). Each downstream forwarder node extracts r from the received Data packet. It compares r to the link rate $R(t)$ that has been pre-calculated for the link through which the Data packet has been received. If $R(t)$ is smaller than r, the forwarder updates ($r = R(t)$. When finally a consumer receives a Data packet, it knows the bottleneck rate of the traversed path. Now, the consumer assigns this value as the Interest sending rate. MIRCC employs a multipath forwarding strategy to make use of all available network resources. Compared to RCP, the simulation results prove a faster convergence

time with less overshoot and oscillation.

Carofiglio et al. [37] design a set of optimal dynamic multipath congestion control schemes from a multi-commodity flow problem. They also propose a Request Forwarding Algorithm (RFA). RFA keeps track of the PIT entries for each content prefix and each face. A face's forwarding probability is estimated from a weight. This weight is a moving average over the reciprocal count of the PIT entries.

Carofiglio et al. [38] propose a hop by hop Interest shaping mechanism. It enables per-flow Interest rate control at each NDN router's output interface. The authors calculate each NDN router's available capacity and use Interest shaping to dynamically update their data rate and transmission buffer occupancy. They also keep one virtual queue per flow in each output interface. Each virtual queue has a credit counter with a maximum value B (in bytes) that indicates how many Data bytes the flow is allowed to send without any additional delay.

Abu et al. [39] present a congestion control approach that is based on PIT occupancy and data transmission buffer per interface. PIT occupancy is used to estimate the anticipated data buffer occupancy. When the anticipated data buffer occupancy is greater than a certain threshold, the Data packets are marked. In addition to that packet, marking is done by applying the Random Early Detection (RED) scheme in PIT. After receiving a marked data packet, the consumer adjusts the interest window size using the AIMD approach. This scheme inherits the problem of parameter tuning due to the application of RED.

Gündoğan et al. [40] discuss PIT's overall effect on network performance along with link capacity.

PCON [41] uses CODEL AQM scheme [33] from TCP/IP. CODEL approach considers queuing delay for congestion indication, and this parameter is independent of RTT or link rate. After congestion detection, the router issues a signal to consumers by explicitly marking the Data packets. Upon receiving these marked Data packets, downstream routers redirect the traffic to alternative paths. In the case of consumers, they lower their Interest sending rates. However, PCON has not considered the Pending Interest Table's finite size. In addition to that, they have not considered the presence of non-responsive consumers.

To the best of our knowledge, most of the existing work except [39, 40] have not considered the existence of PIT. Furthermore, no one has considered the presence of non-responsive consumers.

5.3 Rationale behind considering PIT per outgoing face placement

This section explains our motivation behind considering PIT per outgoing face placement. Please note that, in our previous works, we have considered a single PIT per node. Due to shifting from one PIT per node to distributed PIT (PIT per outgoing face), the packet forwarding process becomes different from the conventional NDN packet forwarding process. We have already discussed the packet forwarding for single PIT in section 2.1.5 in chapter 2. Here, we discuss the packet forwarding process in PIT per outgoing face placement and how we have implemented it in NFD.

5.3.1 Motivation behind considering PIT per outgoing face placement

We consider Figure 5.1 as an example of illustration to discuss our motivation for considering PIT per outgoing face placement. As we can observe in Figure 5.1, there are two consumers, C1 and C2. They send Interests with name prefixes '/prefix/A' and '/prefix/B', respectively. There are three producers, P1, P2 and P3, producing contents with prefixes '/prefix/A', '/prefix/B' and '/prefix/B', respectively. We assume R4-R5 as a bottleneck link. We consider C2 as a responsive consumer who sends Interests following a congestion control policy. On the other hand, C1 is a non-responsive consumer which sends bursty traffic (we consider 1000 Interests/sec in our experiment). Below, we explain the benefits of Distributed PIT (PIT per outgoing face) over a single PIT. For that, we discuss both cases one by one.

Case 1: Single PIT (refer to Figure 5.1)

Due to bottleneck link R4-R5, the traffic burst from C1 creates congestion at router R4. Due to this, Interests from C1 and C2 are satisfied with more delay. It increases the PIT occupancy, and in less time PIT may reach its full capacity. The new incoming Interests from C2 get dropped as there is no space in PIT. Due to central PIT, even though there are available paths for C2's Interests through R4-R7 and R4-R6, we can not forward through these paths. This shows the limitation of a single PIT. In addition to that, due to packet dropping, C2 experiences time out and reduces its congestion window. Thus C2's throughput is highly affected.

It seems that we can easily address this issue by reserving PIT entries for each name prefix or distribution of PIT entries across all incoming interfaces. However, these solutions are ineffective as Interests from C1 can still occupy some portions of PIT.

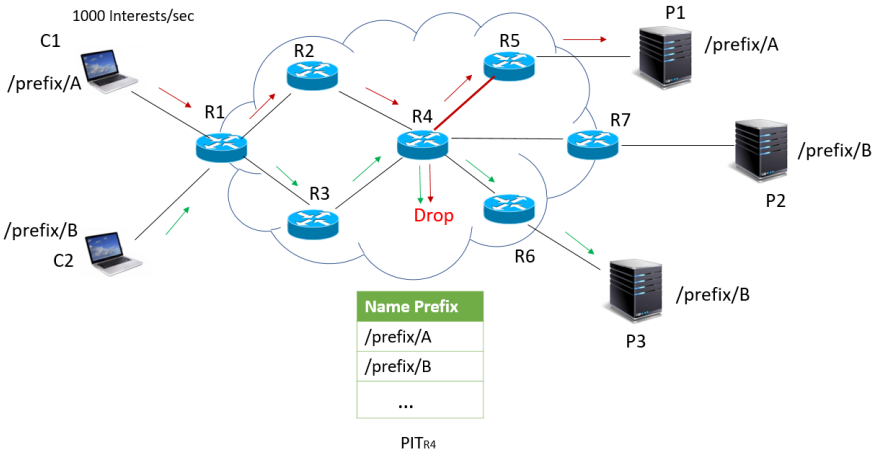


Figure 5.1: Single PIT

- **Case 2: Distributed PIT (refer to Figure 5.2)**

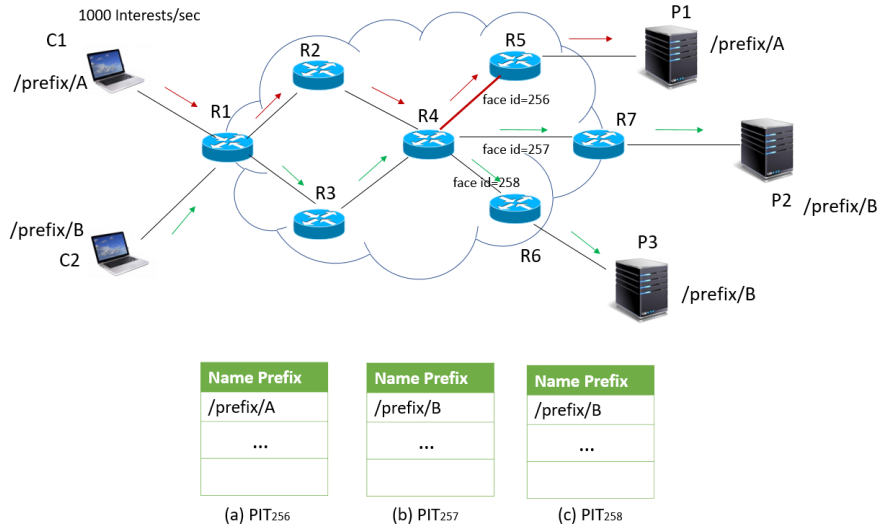


Figure 5.2: Distributed PIT (PIT per outgoing interface)

In this case, PIT is placed in each outgoing interface of a router. In Figure 5.2, there are three PITs with outgoing face ids 256, 257, and 258 in router R4. They are named as PIT_{256}, PIT_{257} and PIT_{258} for convenience. Interests from C1 are entered into PIT_{256}. On the other hand, due to PIT per outgoing face placement, C2's Interests can be for-

warded through either R4-R6 or R4-R7. Therefore, traffic bursts from C1 can not impact C2's performance.

Another benefit of considering PIT per outgoing face placement is that it also helps to estimate PIT size compared to a single PIT.

- ***PIT size Estimation:***

  PIT per outgoing face helps to estimate PIT size as follows:

$$PS = [BW/(l_I + l_I * k * \rho] * [(\tau - \delta_{rtt}) * (1 - \rho) + \delta_{rtt}] \tag{5.1}$$

  where PS is the PIT size, BW is the bandwidth of the outgoing interface, l_I is the average Interest packet size, k is the ratio between Data and Interest packet sizes, ρ is the ratio of well behaved Interest traffic, τ is PIT entry timeout (default value: 4sec), δ_{rtt} is the average round trip time. We adopt Equation (5.1) from [42].

5.3.2 Forwarding of NDN packets in PIT per outgoing face placement

Our implementation of PIT per outgoing face placement in NFD is shown in Algorithm 10. Upon the arrival of an Interest, the router first looks up at CS for a satisfying Data packet. If it is successful, it returns the Data packet to the interface through which Interest has been received. Otherwise, it examines FIB for appropriate outgoing interface (refer to 5.3). Please note that we maintain a distinct PIT table *m_pit* for Interests with the name prefix '/localhost/'. The reason for considering a separate table is that these Interests are *control command requests*. A router does not forward these Interests to a non-local face; instead, they are forwarded to a specific face named *Internal face* [6]. A list named Map is maintained at each router. Map contains a list of outgoing face IDs and pointers to their corresponding PIT tables. Since we are considering PIT per outgoing placement, one PIT is placed at each outgoing interface of the router. So, pointers to these PIT tables are maintained in a dynamic array named *pitList*.

Suppose an Interest arrives, its outgoing interface is selected after looking up at FIB. However, the PIT is already full. In that situation, the router sends a NACK with reason code 'PIT FULL' (refer to line number 16 in Algorithm 10). Reason codes must be included so that downstream routers can figure out why the upstream router has delivered the NACK. So, in this case, if the downstream routers receive a NACK, it extracts the reason code and finds it to

be 'PIT FULL', then it diverts future Interests to another path. A PIT entry is created to the selected outgoing interface. The information of incoming and outgoing interface is also stored in PIT. Then the Interest packet is forwarded through this outgoing face.

After the arrival of a Data packet, the router first checks whether this Interest has a /'localhost/' prefix. If yes, it looks up at PIT table *m_pit* to forward this Interest. Otherwise, the router looks up at PIT corresponding to that interface (by looking at *Map*). The incoming Data packet is stored in CS according to CS's admission policy for future requests. After that, the Data packet is forwarded to the incoming interfaces listed in the PIT entry, and finally, the PIT entry is removed.

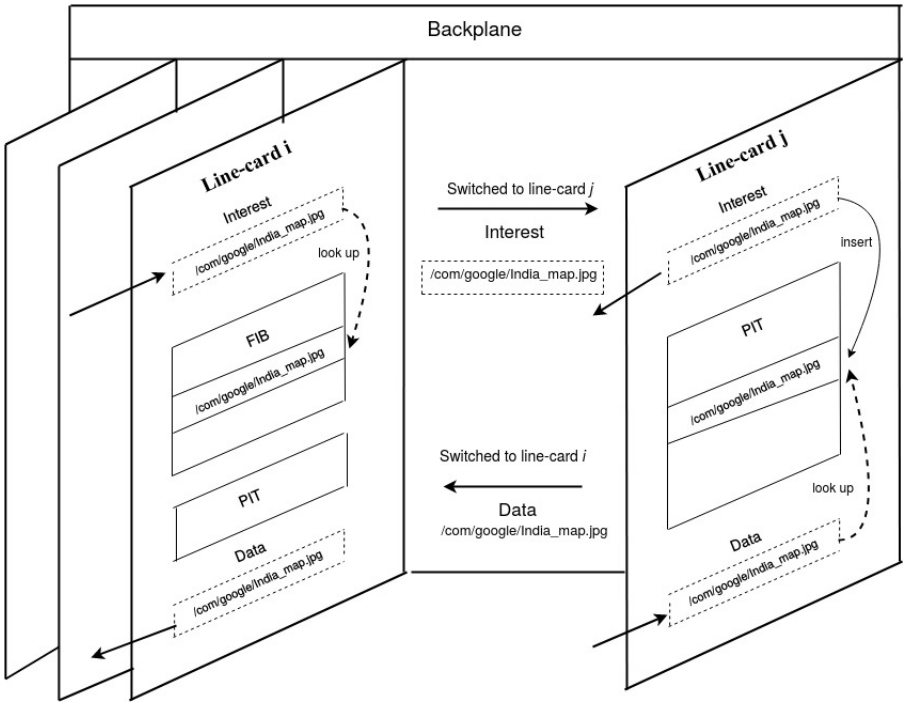


Figure 5.3: Packet forwarding in PIT per outgoing face placement (adopted from [14]), the presence of CS is omitted to reduce complexity

Algorithm 10: NDN packet processing at router

```
   // please refer to Table 5.1 for details of the variables used in this Algorithm
 1 function OnIncomingInterest ( Interest, inFace )
 2 Name=Interest.getName();
 3 if Data ← CS.Find(Name) then
 4     return Data
 5 else
 6     if Name.Prefix("/localhost/") then
 7         m_pit.insert (Interest)
 8     else
 9         fibEntry= FIB. find(Interest. Name) ;
10         for each interface in fibEntry do
11             outFace = nexthop.getFace()FaceId=outFace.getId()
12             if Map.Find (FaceId)== Map.end() then
13                 Map[FaceId]= pitList[c++]
14             else
15                 if Map[FaceId] → size() ≤ MaxSize_FaceId then
16                     Drop Interest and send NACK with reason code 'PIT FULL'
17                     return
18             pitEntry= Map[FaceId] → insert (interest) ;
19             pitEntry.incomingFace= inFace; pitEntry.outgoingFace= outFace
20             Forward Interest to outFace ;

21 function OnIncomingData (Data, inFace2 )
22 dataName=Data.getName(); FaceId=inFace2.getId() ;
23 if dataName.Prefix("/localhost/") then
24     pitMatches= m_pit.Find ( dataName) ;
25 else
26     pitMatches=Map[FaceId] → Find( dataName) ;
27     CS.Insert (Data) ;
28 for each pitEntry in pitMatches do
29     Forward Data to all incoming faces of pitEntry ;
30     Delete pitEntry ;
```

Table 5.1: Different Variables used in Algorithm 10

Field	Description
$inFace$	Incoming interface of Interest packet
$inFace2$	Incoming interface of Data packet
$MaxSize_{FaceId}$	Maximum PIT size of PIT table corresponding to face $FaceId$ according to Equation (5.1)
m_pit	A PIT table corresponding to Internal face
$pitList$	A dynamic array of pointers of the PIT tables corresponding to the outgoing interfaces
Map	A list of mapping of outgoing face Ids and their corresponding PIT table pointers
$pitMatches$	A list of matched PIT entries
c	A variable (initial value is 0)

5.4 Proposed Scheme: LPECN

There are three major elements in LPECN.

- *Congestion detection and signalling:* Each router detects congestion by monitoring PIT size and queuing delay. In case of PIT becomes full, the router sends a NACK with reason code "PIT FULL" so that downstream routers can divert the traffic to another route. On the other hand, if the estimated queuing delay exceeds a pre-determined threshold, the router marks the outgoing Data packets to inform the consumers to reduce the Interest sending rate.

- *Adaptation of Consumer window:* Consumers respond to congestion by adjusting their consumer window after receiving NACK or marked Data packets.

- *Forceful Interest rate limitation:* After receiving a NACK or congestion marking, Consumer-side edge (CE) routers compare the incoming Interest rates for current and previous time intervals. If the rates are still equal, CE routers consider that Interests are received from non-responsive (NR) consumers. After detection of NR consumers, CE routers restrict the number of PIT entries for these consumers. The number of allowable PIT entries is

estimated based on ISR value per name prefix per interface.

The details of each element are discussed below.

5.4.1 Congestion detection and Signalling

Our proposed scheme detects congestion by monitoring PIT size and queuing delay. As we can see in Figure 5.4, there are two queues in each router: one for storing forwarded Interests and the other for incoming Data packets. In Chapter 3, we have already discussed the importance of limitation of PIT size. PIT size is limited using Equation (5.1). When a router receives an Interest, first, it finds the PIT (by looking up at FIB).

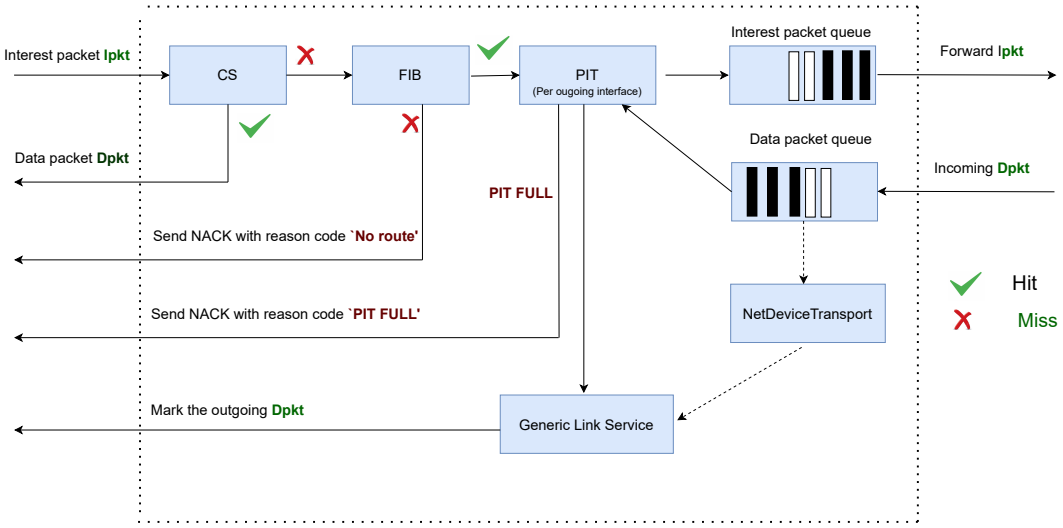


Figure 5.4: Congestion detection and signalling in NDN

If PIT is full, it sends a NACK with the reason code 'PIT FULL'. After receiving this NACK, the routers in the downstream path search for alternative paths. If it becomes successful, it forwards future requests towards these paths. Otherwise, it sends a NACK with reason code 'No Route' towards downstream. If the consumer receives a NACK, it reduces its Interest sending rate. Please note that we can not accurately estimate PIT size. The reason behind it is that PIT size is calculated from RTT and average Data packet size. We can not accurately estimate these values because of the network's dynamic nature. However, the buffer at the outgoing interface smoothens out these fluctuations.

In recent times, most applications require a shorter content retrieval time. To maintain a lower content retrieval time, we leverage the CODEL AQM approach [33]. CODEL AQM uses queuing delay as the parameter for congestion indication. Queuing delay ('sojourn time') depends neither on Round Trip Time nor link rate. To calculate sojourn time, a timestamp is added as a tag to the NDN packet during enqueue of the packet. Sojourn time is calculated as time::now () - tag.GetEnqueuTime() (refer to line number 7 of Algorithm 11). Later, during dequeue, GetEnqueuTime() function extracts the enqueue timestamp from the NDN packet.

Table 5.2: Different Variables used in Algorithm 11

Field	Description
$packt$	Incoming NDN packet (link layer)
$pktQueue$	A queue of packets
$flag$	A variable (initial value false).
$MarkingInterval$	Congestion marking Interval
$TargetDelay$	Target Queue delay
$sojourn$	Amount of time spent by a packet in a buffer
$NextMarkTime$	Time to mark next packet (initial value= time::max())
$NoMarkedCurrent$	The number of marked Data packets in current congestion

A variable named $flag$ is maintained for the detection of congestion. Router compares *sojourn time* and target delay (default value: 5msec). If the value of sojourn time exceeds the target delay, we set the value of the $flag$ to 1. When a link-layer Data packet arrives at a router, before forwarding it downstream, the Generic Link service Module monitors the value of $flag$ (refer to line number 15 of Algorithm 11). It access the value of $flag$ from NetDeviceTransport module by using $getTransport()$ function. If the value of the $flag$ is 1 for predetermined congestion marking Interval (default: 100msec), then the outgoing Data packet is marked. The marking is done by setting 'congestionMarkingfield' of the outgoing Data packet. After receiving a Data packet, a consumer first extracts the 'congestionMarkingfield' value. If the value is 1, the consumer reduces the congestion window depending on the congestion control policy.

Algorithm 11 shows how congestion detection and signalling are implemented in NFD.

Algorithm 11: Algorithm for Congestion Detection and Signalling

```
// please refer to Table 5.2 for details of the variables used in this Algorithm
function Queue.DoEnqueue(Ptr < packt >)
CoDelTimestamp tag ;                                   // a tag is added to the packet
tag.EnqueueTime=time::now() packt → AddpacketTag(tag) pktQueue.insert(packet)
function Queue.DoDequeue()
Ptr < packt > packet = pktQueue.delete ()
packt → removePacketTag(tag)
Time sojourn = time :: now() − tag.GetEnqueuTime()
flag=0 ;
if sojourn > TargetDelay then
    flag=1;
return packet ;
function NetDeviceTransport.GetFlagValue()
Ptr < ns3 :: Queue > txQueue=Get < ns3 :: Queue > return txQueue → flag ;

function GenericLinkService.CheckCongestionLevel(dPkt)
bool flag= getTransport() → GetFlagValue() ;        // This function calls function above
if flag! = 1 then
    return ;
else
    if NextMarkTime == time::max() then
        NextMarkTime = now + MarkingInterval
    else if now >= NextMarkTime then
        packt.set < lp :: CongestionMarkField > (1) ;                    // dPkt is marked
        NoMarkedCurrent++ ;
        time::interval= √(MarkingInterval / NoMarkedCurrent+1) ;
        NextMarkTime +=interval ;
    else if NextMarkTime! = time :: max() then
        NextMarkTime = time::max() ;
        NoMarkedCurrent= 0 ;
    else
        ;
```

5.4.2 Adaptation of Consumer window

We have chosen TCP CUBIC [32] congestion control scheme at the consumer side. This is due to CUBIC's stability and scalability over fast and long-distance networks. Algorithm 12 considers CubicIncrease() and CubicDecrease() function straight from CUBIC. We can directly find the implementation of CUBIC in 'ndn-consumer-pcon.cpp' file in ndnSIM simulator (version 2.8).

Algorithm 12: NDN consumer window adaptation

```
1 function onData (data)
2 if data.getCongestionMarks() > 0 then
3 |   CubicDecrease() ;
4 else
5 |   CubicIncrease() ;

6 function onTimeOut (sequenceNumber)
7 CubicDecrease()

8 function onNack (< lp :: Nack > nack)
9 CubicDecrease()
```

5.4.3 Forceful Interest rate limitation

In our proposed scheme, consumer-side Edge (CE) routers are responsible for detecting non-responsive (NR) consumers. After detection, CE router limits the Interests from NR consumers. Now, the question may arise: why do we consider only CE routers (not others)? The reason for considering CE router is that consumers (responsive and non-responsive) may send Interests under the same name prefix. So if an upstream router limits the Interests based on a name prefix, it negatively impacts the responsive consumers.

The formal description of detection and limitation of Interest sending rate from NR consumers is depicted in Algorithm 13. We already discussed in subsection 5.4.1, routers send NACK or marked Data packets for signalling congestion. We add a field named '*isrPrefix*' in the NACK and marked Data packets. This field carries the current value of ISR of the corresponding name prefix of the marked Data packet or NACK. CE routers leverage the ISR value

Algorithm 13: Detection and forceful limitation at Consumer-side Edge routers

Input: isr=0.0 /*Interest Satisfaction Rate */, $prev$=0 /* Interest receiving rate during $(n-1)^{th}$ time Interval */, $current$=0 /* Interest receiving rate during current (n^{th}) time Interval */, $Decision\_Interval$=1 sec /* Monitoring Time Interval for Interest rate */, $Non\_Responsive\_List$=NULL /*A list of name prefixes and corresponding incoming interface id */, $NoPitEntry_{Prefix,fId}$=0/* Allowable PIT entries for ($Prefix$, fId) */

```
1  function onData (data)
2  if data.getCongestionMarks() > 0 then
3      NamePrefix=data.getNamePrefix()
4      isr= data.getISR()          // extracts ISR value from isrPrefix field of Interest packet
5      Simulator::schedule ( Seconds (Decision_Interval), CheckDiffIRate, NamePrefix, isr )

6  function onNack (< lp :: Nack > nack)
7  if nack.reasonCode==PIT FULL then
8      NamePrefix=nack.getNamePrefix()
9      isr= nack.getISR()
10     Simulator::schedule ( Seconds (Decision_Interval), CheckDiffIRate, NamePrefix, isr )

11 function CheckDiffIRate (NamePrefix, isr)
12 inFaceList= PIT.findIncomingFace(NamePrefix)
13 for each interface in inFaceList do
14     faceId= interface.GetId()
15     Router estimates prev and current for (NamePrefix , faceId)
16     if prev==current then
                                          // consumer did not reduce its Interest rate
17         Add NamePrefix and faceId to Non_Responsive_List.
18         Calculate total number of PIT entries (n) for (NamePrefix , faceId)
19         NoPitEntry_{NamePrefix,faceId} = n * isr
```

for restricting the incoming Interests from NR consumers.

After receiving a Data packet, consumer-side Edge (CE) router checks whether the Data packet has a marking or not (refer to *getCongestionMarks()* function in Algorithm 13). If yes, then it further extracts the name prefix of the Data packet using *getNamePrefix()* function.

Then CE router extracts ISR value from *isrPrefix*' field of the Data packet using *getISR()* function. Similarly, after receiving a NACK, CE extracts the reason code of the NACK. If it is 'PIT FULL', then it extracts the name prefix and finds the corresponding ISR value as well. There may be multiple consumers who can send Interests using the same name prefix. As a result, CE router begins to monitor the Interest rate per-prefix per interface. In addition to that, CE router also maintains a timer named $Decision\_Interval$ to decide how long a router should wait before deciding whether or not a consumer is responsive. Due to this, CE router monitors the Interest rate per name prefix for current and previous time intervals. Additionally, a list named $Non\_Responsive\_List$ is maintained to keep track of the Interest name prefix and its corresponding incoming Interface Ids. We can see in lines numbers 5 and 10 in Algorithm 13, after each $Decision\_Interval$, CE router observes whether there is any consumer who has not lowered its Interest sending rate even after receiving congestion marking or NACK. If it finds, then it adds the name prefix and the corresponding incoming Interface Id to the $Non\_Responsive\_List$. We add incoming Interface Id to distinguish consumers who send Interests with the same prefix. For each prefix in the $Non\_Responsive\_List$, the CE router also keeps a $RegressTimer$. This timer determines how long a prefix should be kept in the $NonResponsiveList$. With the arrival of a new NACK or Marking Data, CE router updates the $RegressTimer$. After the expiration of this timer, prefix and its interface Id information is deleted from $Non\_Responsive\_List$. Meanwhile, CE router also computes the number of PIT entries per-prefix per interface (n). For each prefix in the $Non\_Responsive\_List$, it also updates n to $n * isr$. The reason behind the multiplication of n with isr is to proportionally reduce the PIT per name prefix per interface. For example, CE router may receive a NACK with isr=0.5 for prefix '/prefix1' and another NACK with isr=0.9 for different prefix '/prefix2'. Therefore, we would like to drop more Interest packets whose name prefixes have a smaller value of isr.

We can achieve non-responsive consumer detection and Interest rate limitation because of NDN's architectural design features: *stateful forwarding plane* and *symmetric forwarding*. Since each router stores the Interest state information at PIT, so precise details on incoming interface of the received Interests are available. This helps to pinpoint the non-responsive consumer. Thus other consumers who follow congestion control policy are unaffected by non-responsive consumers.

5.5 Performance Analysis

We have implemented LPECN in the ndnSIM simulator (version 2.8) [7]. To assess its efficacy, we have compared LPECN to PCON [41]. The reason for comparison is that both schemes use CODEL AQM scheme [33] for congestion detection. However, the significant difference between them is that PCON does not consider the PIT size and uses a single PIT. On the other hand, LPECN limits the PIT size and uses PIT per outgoing face placement. The system we used during the experiment is OS: Ubuntu 16.04, memory:16 GB, processor: core i7-6700 CPU @3.40GHz.

To evaluate the performance of LPECN, we use a small topology for better understanding (refer to Figure 5.5). There are 4 consumers C1, C2, C3 and C4 send interests with $/prefix1$, $/prefix2$, $/prefix3$ and $/prefix4$ respectively. Producers P1, P2, P3 and P4 produces and serves content with prefixes $/prefix1$, $/prefix2$, $/prefix3$ and $/prefix4$ respectively. R1-R2 is a bottleneck link. The bandwidth of other links is shown in the Figure 5.5. The delay of each link is 20ms. Since we use PIT per outgoing face placement, we set PIT size as shown in Equation (5.1). We consider the size of outgoing buffer equal to bandwidth*delay. C2, C3, and C4 send Interests by following a congestion control mechanism. They adjust their consumer window on receiving a congestion signal. On the other hand, C1 transmits a burst of 1000 Interests/sec and ignores any congestion signal. At t=1 sec, C2, C3, and C4 begin issuing Interests and stop at t=42 sec. Consumer C1 begins at t=10 seconds and ends at t=42 seconds. Table 5.3 contains the simulation parameters.

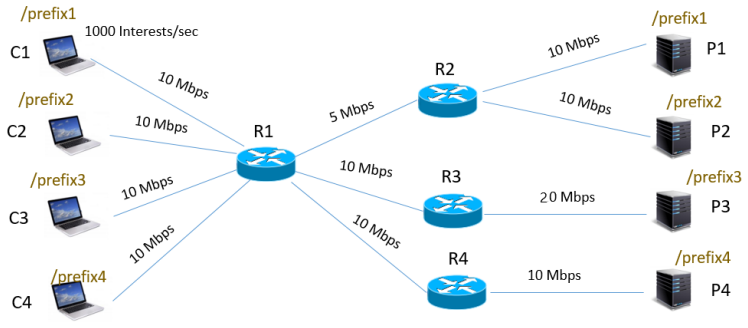


Figure 5.5: Topology

Table 5.3: Simulation Parameters

Parameter	Value
Forwarding Strategy	Best-Route Strategy
CS Replacement Policy	LRU
Interest lifetime	4000 ms
Interest Packet Size	215 Byte
Data Size	1054 Byte
Simulation Time	45 sec

5.5.1 Evaluation Metrics

To assess the performance of LPECN, we consider the following metrics:

(i) *Average Goodput:* The average number of Data packets received per unit time by a consumer.

(ii) *Average Response Time:* The average time gap between sending one Interest and receiving the corresponding Data packet back by consumers.

(iii) *Average Interest Satisfaction Rate:* It is the ratio of the total Data packets received by the consumers to the total Interest packets forwarded.

5.5.2 Simulation results

The goodput at consumer C3 and C4 in both schemes PCON and LPECN are shown in Figure 5.6. The figure shows a rise in goodput up to 10 seconds, then a sharp drop after that. However, we do not observe any decrease in goodput in LPECN. Please note that PCON uses a single PIT per node, and LPECN uses PIT per outgoing face placement.

In case of PCON, at t=10 sec, C1 starts to send a traffic burst. Due to R1-R2 bottleneck link, router R1 experiences congestion, so PIT entries are satisfied gradually, making a large number of entries in PIT and eventually making it full. Once the PIT becomes full, Interests from C3 and C4 are dropped at router R1. Due to packet dropping, C3 and C4 experience packet timeout. In response to this, consumers adjust their congestion window according to the assigned congestion control policy. Finally, it reduces the goodput of the consumers C3 and C4.

In the case of LPECN, node R1 has three outgoing interfaces, R1-R2, R1-R3 and R1-R4, respectively. Since we consider PIT per outgoing face placement, router R1 has three PITs PIT_{R1-R2}, PIT_{R1-R3} and PIT_{R1-R4}. The subscript in the notation denotes the link. Interests from C3 and C4 are entered to PIT_{R1-R3} and PIT_{R1-R4}. So, traffic bursts from C1 can not impact the goodput of C3 and C4.

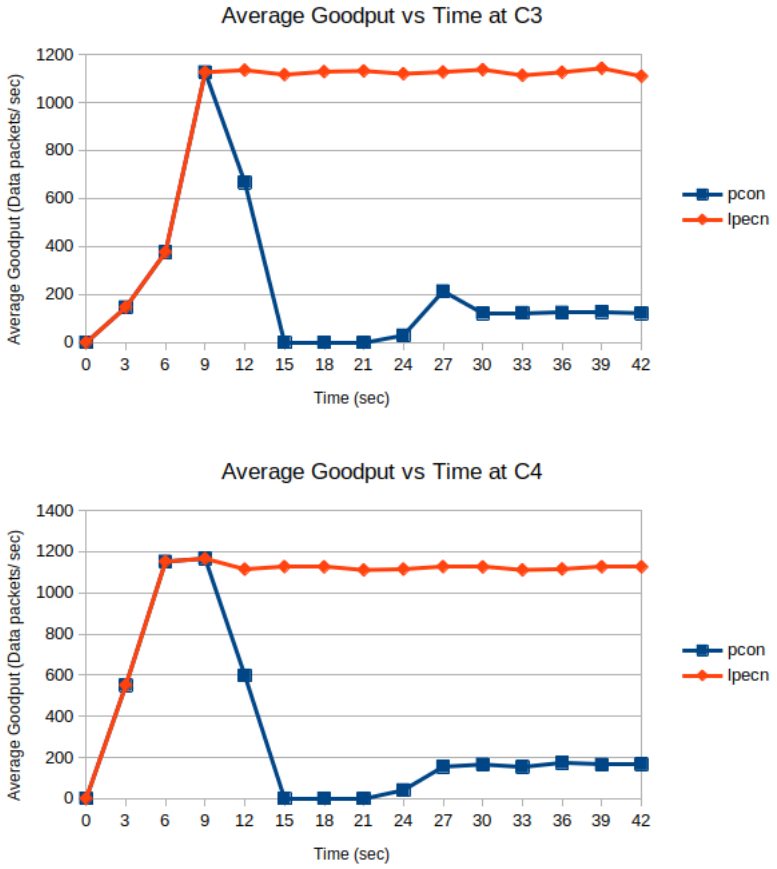


Figure 5.6: Goodput at consumer C3 and C4 in PCON and LPECN scheme

Figure 5.7 demonstrates the Average Response time at consumers C3 and C4 in PCON and LPECN scheme. We do not observe any significant difference in response time for both schemes. It is because both schemes leverage CODEL AQM schemes for controlling queuing delay. In the experiment, we have set the threshold for the estimated queuing delay, which we name as *target delay* to its the default value (5 msec).

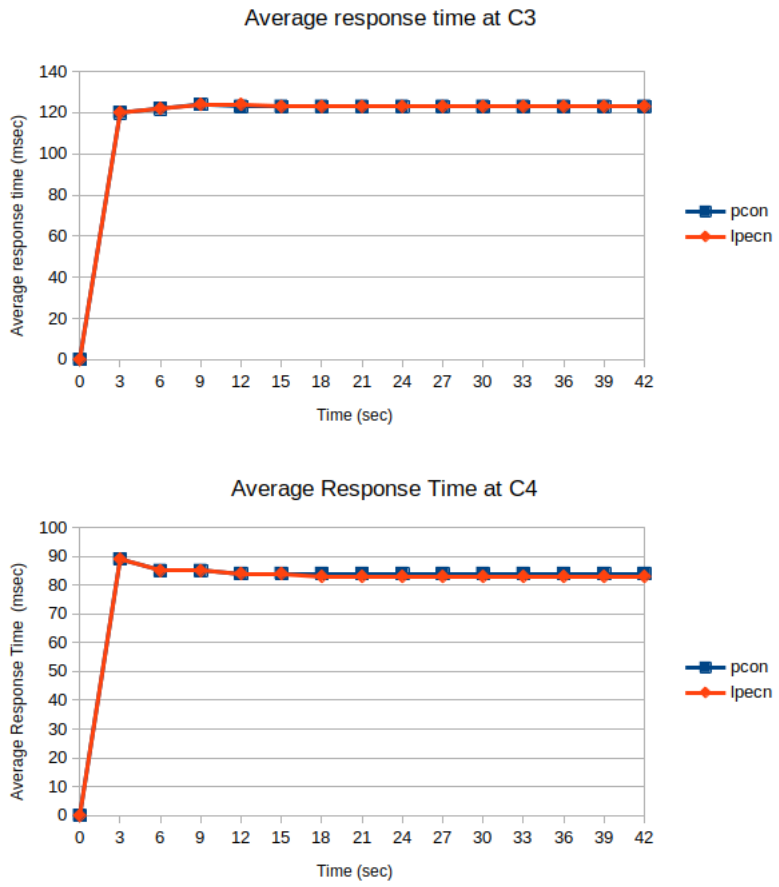


Figure 5.7: Average Response time at consumer C3 and C4 in PCON and LPECN scheme

The Average Interest Satisfaction Rate (ISR) at consumers C3 and C4 are depicted in Figure 5.8. As we discussed earlier, there is no packet drop in case of LPECN due to PIT per outgoing placement. So, LPECN provides satisfactory ISR value for both the consumers. On the other hand, due to single PIT, packets are dropped at R1. So, there are no corresponding Data packets for those dropped Interests. So, ISR falls miserably in case of PCON scheme.

The average Goodput, Response Time and ISR at consumer C2 in PCON and LPECN scheme are shown in Figure 5.9. In the case of PCON, the bursty traffic from C1 creates congestion at R1, resulting in PIT filling up. This causes packet dropping from C2. Packet dropping results in sending NACK. After receiving NACK, consumer C2 decreases its congestion window. It reduces goodput in the case of PCON. However, LPECN detects and limits Interests

from C1. Therefore, though initially, goodput decreases at t= 10 sec; however, later, it maintains goodput because of detection and limitation. Due to congestion in the PCON scheme, response time increases due to an increase in queuing delay. However, response time is constant in LPECN as it applies forceful Interest limitation. Due to packet dropping, there is a significant ISR drop at C2 in PCON as compared to LPECN.

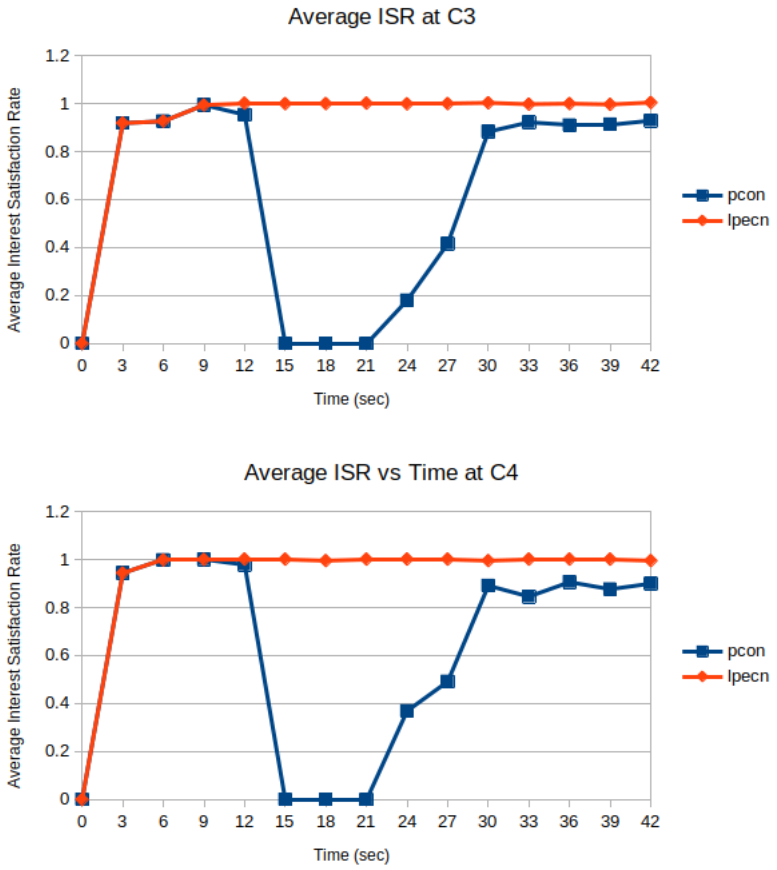


Figure 5.8: Average Interest Satisfaction Rate at consumer C3 and C4 in PCON and LPECN scheme

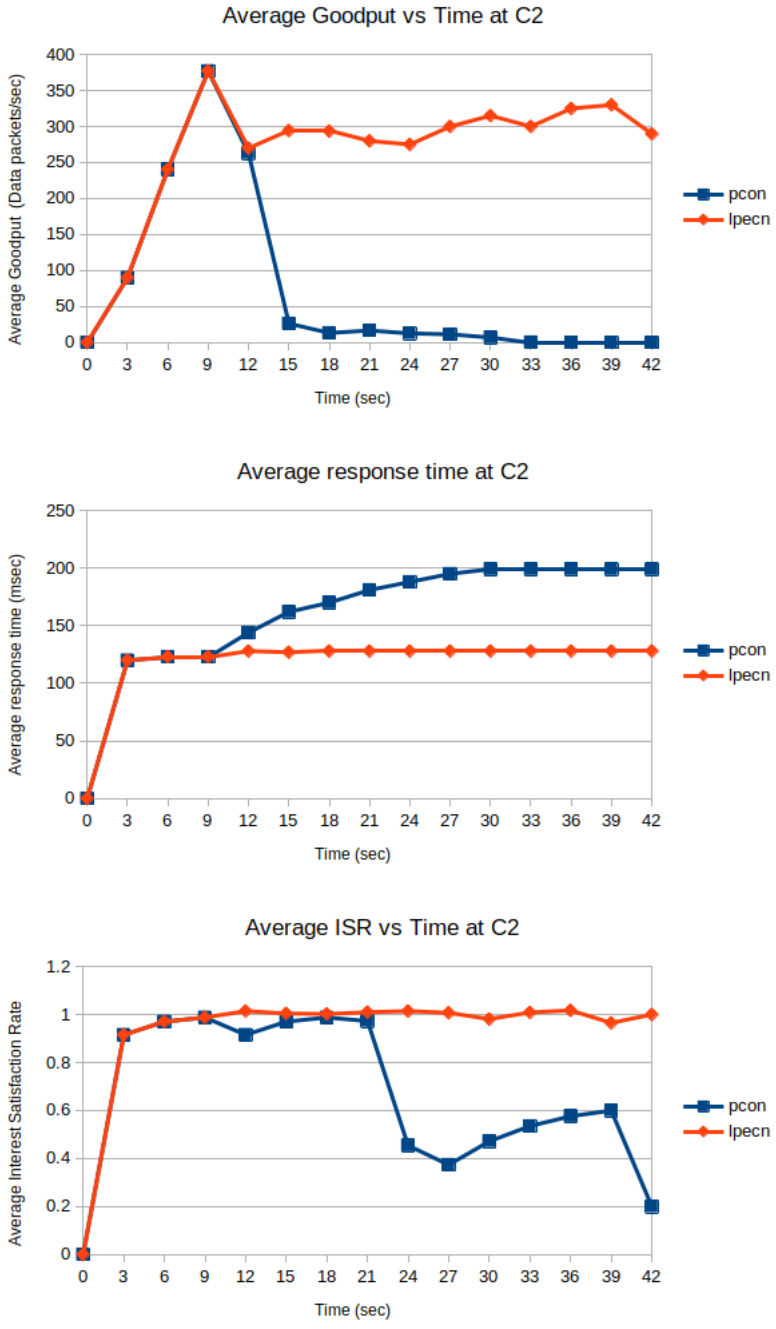


Figure 5.9: Average Goodput, Response Time and ISR at consumer C2 in PCON and LPECN scheme

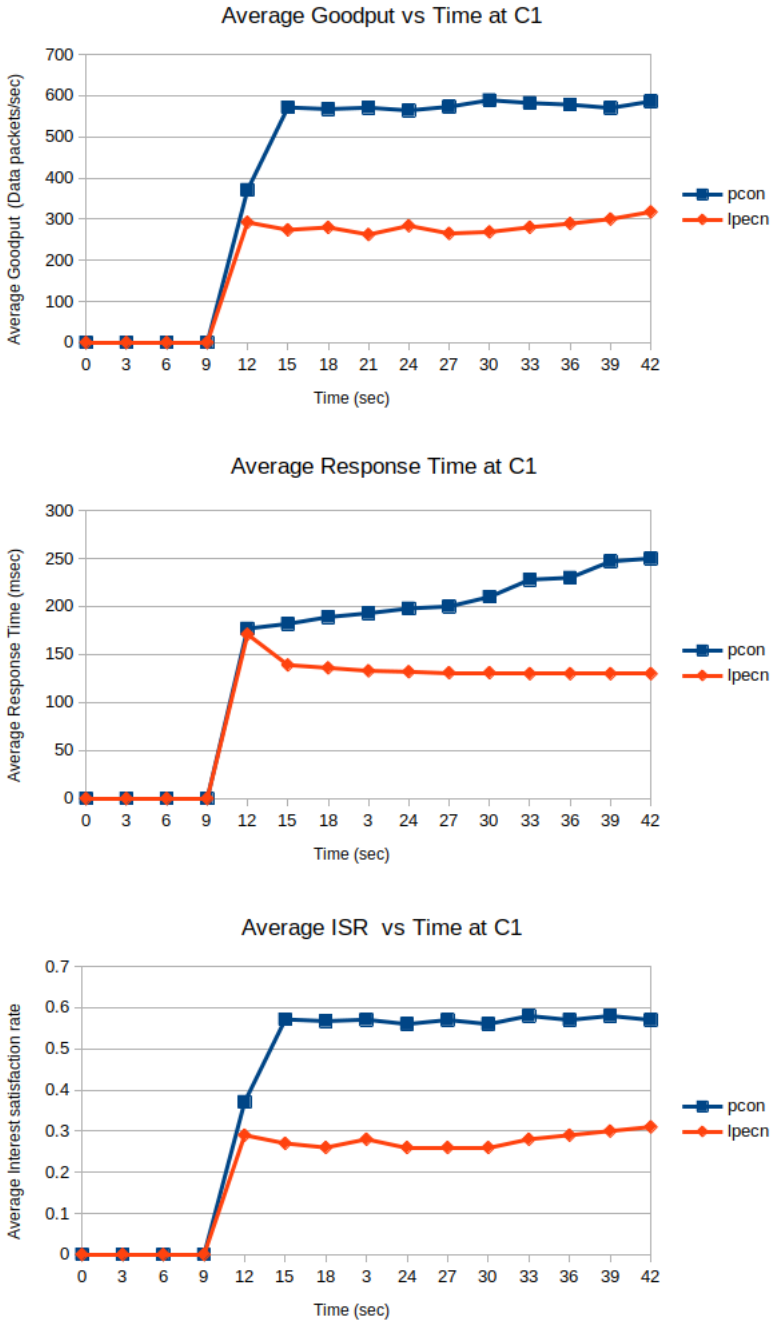


Figure 5.10: Average Goodput, Response Time and ISR at consumer C1 in PCON and LPECN scheme

The average Goodput, Response Time and ISR at consumer C1 for both schemes are depicted in Figure 5.10. Here, we observe an exciting result. PCON shows better results as compared to LPECN for all performance metrics. The reason is that LPECN detects and limits Interests from non-responsive (NR) consumers. On the other hand, PCON does not handle NR consumers. Therefore, because of this limitation, LPECN has a lower value in goodput and ISR as compared to PCON.

In case of LPECN, consumer-side edge router R1 detects C1 as a non-responsive consumer. So, it forcefully limits Interests from C1. This decreases the number of packets in the outgoing queue, reducing the queuing delay. On the other hand, PCON does not apply any limitation on C1, which increases the number of packets in the outgoing queue and increases the queuing delay. So, in the case of LPECN, the response time decreases.

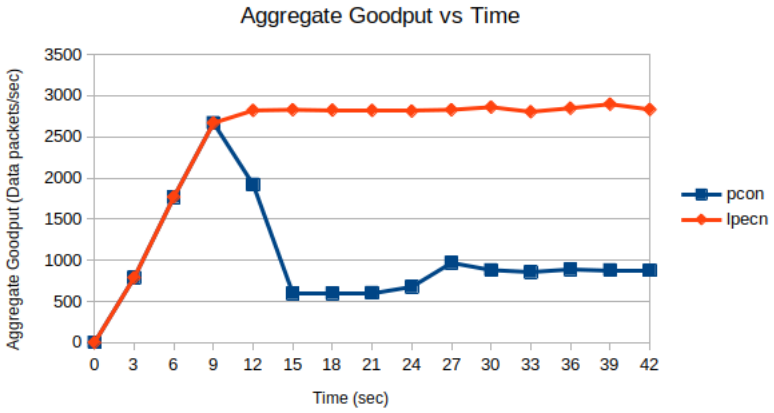


Figure 5.11: Aggregate Goodput in PCON and LPECN scheme

The aggregate goodput of all consumers for PCON and LPECN scheme is shown in Figure 5.11. As we can see in the Figure, LPECN has a substantially larger goodput than the PCON.

5.6 Summary

In this chapter, we have demonstrated the effectiveness of PIT per outgoing face placement from congestion control perspective. Because of NDN's architectural elements: *stateful forwarding plane* and *symmetric forwarding*, we could detect and limit Interests from non-responsive consumers by utilizing NACK and congestion marking. Simulation studies show that LPECN can

successfully handle non-responsive consumer interests while ensuring that those who follow the congestion control strategy are unharmed.

Intentionally Left Blank

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

In this thesis, we have investigated the impact of PIT size on network performance from three different perspectives: QoS, security and congestion control. In this direction, the major contributions of this thesis are summarized as follows.

In Chapter 3 (first contribution), we discuss the importance of limiting the PIT size. We observed that, during traffic burst, PIT may become full. It reduces the QoS of the premium consumers. To address this problem, we propose two schemes: PRWR (PIT Replacement Without Reservation) and PRR (PIT Replacement with Reservation) to enhance QoS. We present analytical models of both schemes using two-dimensional CTMC. We compute the performance metrics, namely non-prioritized Interest blocking probability and non-prioritized forced termination probability. The simulation and numerical results show that PRR gives better performance as compared to the PRWR scheme at the marginal cost in terms of non-prioritized blocking probability.

PIT can also be exploited by attackers to launch an attack. Attackers can send a massive number of Interests to fill up the PIT of the intermediate routers. As a result, subsequent Interest packets are dropped. In Chapter 4, we developed a smart collaborative attack model to degrade the QoS of the targeted legitimate consumers. In a collaborative attack, malicious consumers send requests for contents that are served by a malicious producer. The attack is designed in such a way that it dynamically sets the Interest rate so that there is no significant

packet drop. This helps in avoiding detection by routers.

The work proposed in Chapter 5 shows the benefits of PIT per outgoing placement from a congestion control perspective. We have used PIT size and queuing delay for congestion detection. The proposed design also considers the presence of non-responsive consumers. We also show that NDN's design features: stateful forwarding plane, and symmetric forwarding help to detect and limit Interests from non-responsive consumers.

6.2 Future Directions

Some of the future directions of this thesis are described as follows.

- **Efficient countermeasure for SCAN attack:** In this thesis, we investigated the SCAN attack, where attackers dynamically change their Interest sending rate depending on network behaviour. After observing the feasibility of such type of sophisticated attack (from simulation experiments), we plan to design its countermeasure in our future research. The solution should be lightweight and efficient to provide a good trade-off between solution cost and benefit. Furthermore, along with SCAN, other attacks such as cache poisoning and cache pollution can simultaneously occur in the network. For example, malicious producers in the SCAN attack model can also poison other caches in the network by providing poisoned content. We can already find mitigation schemes for cache poisoning/cache pollution attacks in the literature. So, it seems that we can combine mitigation schemes for individual attacks and implement them in each NDN node to defend against these attacks. However, this approach may cause significant overhead. Therefore, it seems more reasonable to design a unified solution framework that considers all the possible attacks in NDN. We plan to explore the solution space in the following possible directions: machine learning-based approach, application of trust mechanism among different NDN nodes.

- **Evaluation of fairness of LPECN:** Fairness is explored extensively in current TCP/IP architecture. In general, it is described in terms of flow. However, with the shift in the communication model from point-to-point to point-to-multipoint, it becomes necessary to redefine the term *flow* from the NDN perspective. In NDN, we receive contents from multiple producers using multiple paths, complicating the flow concept. To simplify it, we can evaluate fairness from different perspectives, such as user fairness, content fairness,

link/interface fairness, and flow fairness. According to Oueslat et al. [43], a flow consists of packets having the same object name. As per the definition of MIRCC [35], a flow is a collection of Interests issued by a single consumer, all of which have the same prefix and may be served by one or more producers. So, as future work, we need to extensively investigate each kind of fairness. Along with that, we also consider the evaluation of the fairness of LPECN as our future work.

- **Enhancing Quality of Experience (QoE) of consumers:** Nowadays, to increase QoE of the consumers, the congestion control policy should work in conjunction with other QoS mechanisms such as traffic conditioning, queue scheduling so that combined output can help us to provide less delay, minimum packet loss and more bandwidth according to SLA agreement. To design efficient queue scheduling mechanism in NDN, a router can consider the applications' delay requirements that is explicitly defined in the Interest packet itself. In future, we aim to explore new queue scheduling mechanisms in NDN.

Intentionally Left Blank

Bibliography

[1] VNI Cisco. "Cisco visual networking index: Forecast and trends, 2017–2022 white paper". **in**: *Cisco Internet Report* 17 (2019), **page** 13.

[2] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros **and** George C Polyzos. "A survey of information-centric networking research". **in**: *IEEE communications surveys & tutorials* 16.2 (2013), **pages** 1024–1049.

[3] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang **and** Beichuan Zhang. "Named data networking". **in**: *ACM SIGCOMM Computer Communication Review* 44.3 (2014), **pages** 66–73.

[4] Yalei Tan, Qing Li, Yong Jiang **and** Shutao Xia. "Rapit: Rtt-aware pending interest table for content centric networking". **in**: *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE. 2015, **pages** 1–8.

[5] Madhurima Buragohain, Prashant Gudipudi, Md Zaki Anwer **and** Sukumar Nandi. "EQPR: enhancing QoS in named data networking using priority and RTT driven PIT replacement policy". **in**: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, **pages** 1–7.

[6] Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto **andothers**. "NFD developer's guide". **in**: *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021* (2014).

[7] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko **and** Lixia Zhang. "ndnSIM 2: An updated NDN simulator for NS-3". **in**: *NDN, Technical Report NDN-0028, Revision 2* (2016).

[8] John Stanik. "A Conversation with Van Jacobson: The TCP/IP pioneer discusses the promise of content-centric networking with BBN chief scientist Craig Partridge." **in**: *Queue* 7.1 (2009), **pages** 8–16.

[9] Haowei Yuan, Tian Song **and** Patrick Crowley. "Scalable NDN forwarding: Concepts, issues and principles". **in**: *2012 21st International Conference on computer communications and networks (ICCCN)*. IEEE. 2012, **pages** 1–9.

[10] Haowei Yuan **and** Patrick Crowley. "Scalable pending interest table design: From principles to practice". **in**: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE. 2014, **pages** 2049–2057.

[11] Wei You, Bertrand Mathieu, Patrick Truong, Jean-François Peltier **and** Gwendal Simon. "Dipit: A distributed bloom-filter based pit table for ccn nodes". **in**: *2012 21st International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2012, **pages** 1–7.

[12] Zhuo Li, Kaihua Liu, Yang Zhao **and** Yongtao Ma. "MaPIT: an enhanced pending interest table for NDN with mapping bloom filter". **in**: *IEEE Communications Letters* 18.11 (2014), **pages** 1915–1918.

[13] Huichen Dai, Jianyuan Lu, Yi Wang **and** Bin Liu. "BFAST: Unified and scalable index for NDN forwarding architecture". **in**: *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2015, **pages** 2290–2298.

[14] Huichen Dai, Bin Liu, Yan Chen **and** Yi Wang. "On pending interest table in named data networking". **in**: *2012 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE. 2012, **pages** 211–222.

[15] Matteo Varvello, Diego Perino **and** Leonardo Linguaglossa. "On the design and implementation of a wire-speed pending interest table". **in**: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2013, **pages** 369–374.

[16] Bastiaan Wissingh, Christopher A Wood, Alexander Afanasyev, Lixia Zhang, David Oran **and** Christian F Tschudin. "Information-Centric Networking (ICN): Content-Centric Networking (CCNx) and Named Data Networking (NDN) Terminology." **in**: *RFC* 8793 (2020), **pages** 1–17.

[17] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang **and** Lixia Zhang. "A case for stateful forwarding plane". **in**: *Computer Communications* 36.7 (2013), **pages** 779–791.

[18] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D Thornton, Diana K Smetters, Beichuan Zhang, Gene Tsudik, Dan Massey, Christos Papadopoulos **andothers**. "Named data networking (ndn) project". **in**: *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC* 157 (2010), **page** 158.

[19] Suhaidi Hassan, Raaid Alubady **and** Adib M Monzer Habbal. "Performance evaluation of the replacement policies for pending interest table". **in**: *Journal of Telecommunication, Electronic and Computer Engineering* 8.10 (2016), **pages** 125–131.

[20] Raaid Alubady, Suhaidi Hassan **and** Adib Habbal. "HLLR: highest lifetime least request policy for high performance pending interest table". **in**: *2016 IEEE Conference on Open Systems (ICOS)*. IEEE. 2016, **pages** 42–47.

[21] Lorenzo Saino, Cosmin Cocora **and** George Pavlou. "CCTCP: A scalable receiver-driven congestion control protocol for content centric networking". **in**: *2013 IEEE international conference on communications (ICC)*. IEEE. 2013, **pages** 3775–3780.

[22] Giovanna Carofiglio, Massimo Gallo **and** Luca Muscariello. "ICP: Design and evaluation of an interest control protocol for content-centric networking". **in**: *2012 Proceedings IEEE INFOCOM Workshops*. IEEE. 2012, **pages** 304–309.

[23] Andriana Ioannou **and** Stefan Weber. "A survey of caching policies and forwarding mechanisms in information-centric networking". **in**: *IEEE Communications Surveys & Tutorials* 18.4 (2016), **pages** 2847–2886.

[24] Paolo Gasti, Gene Tsudik, Ersin Uzun **and** Lixia Zhang. "DoS and DDoS in named data networking". **in**: *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2013, **pages** 1–7.

[25] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun **and** Lixia Zhang. "Interest flooding attack and countermeasures in named data networking". **in**: *2013 IFIP Networking Conference*. IEEE. 2013, **pages** 1–9.

[26] André Nasserala **and** Igor Monteiro Moraesy. "Analyzing the producer-consumer collusion attack in Content-Centric Networks". **in**: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2016, **pages** 849–852.

[27] Eslam G AbdAllah, Hossam S Hassanein **and** Mohammad Zulkernine. "A survey of security attacks in information-centric networking". **in**: *IEEE Communications Surveys & Tutorials* 17.3 (2015), **pages** 1441–1454.

[28] Alberto Compagno, Mauro Conti, Paolo Gasti **and** Gene Tsudik. "Poseidon: Mitigating interest flooding DDoS attacks in named data networking". **in**: *38th annual IEEE conference on local computer networks*. IEEE. 2013, **pages** 630–638.

[29] Hani Salah **and** Thorsten Strufe. "Evaluating and mitigating a collusive version of the interest flooding attack in NDN". **in**: *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE. 2016, **pages** 938–945.

[30] Neil Spring, Ratul Mahajan **and** David Wetherall. "Measuring ISP topologies with Rocketfuel". **in**: *ACM SIGCOMM Computer Communication Review* 32.4 (2002), **pages** 133–145.

[31] Benoit Donnet **and** Timur Friedman. "Internet topology discovery: a survey". **in**: *IEEE Communications Surveys & Tutorials* 9.4 (2007), **pages** 56–69.

[32] Sangtae Ha, Injong Rhee **and** Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant". **in**: *ACM SIGOPS operating systems review* 42.5 (2008), **pages** 64–74.

[33] Kathleen Nichols, Van Jacobson, Andrew McGregor **and** Jana Iyengar. "Controlled delay active queue management". **in**: *RFC 8289* 1 (2018), **pages** 1–25.

[34] Yaogong Wang, Natalya Rozhnova, Ashok Narayanan, David Oran **and** Injong Rhee. "An improved hop-by-hop interest shaper for congestion control in named data networking". **in**: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), **pages** 55–60.

[35] Milad Mahdian, Somaya Arianfar, Jim Gibson **and** Dave Oran. "MIRCC: Multipath-aware ICN rate-based congestion control". **in**: *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 2016, **pages** 1–10.

[36] Nandita Dukkipati. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. Citeseer, 2008.

[37] Giovanna Carofiglio, Massimo Gallo **and** Luca Muscariello. "Optimal multipath congestion control and request forwarding in information-centric networks: Protocol design and experimentation". **in**: *Computer Networks* 110 (2016), **pages** 104–117.

[38] Giovanna Carofiglio, Massimo Gallo **and** Luca Muscariello. "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks". **in**: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), **pages** 491–496.

[39] Amuda James Abu, Brahim Bensaou **and** Ahmed M Abdelmoniem. "Inferring and controlling congestion in CCN via the pending interest table occupancy". **in**: *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE. 2016, **pages** 433–441.

[40] Cenk Gündoğan, Jakob Pfender, Michael Frey, Thomas C Schmidt, Felix Shzu-Juraschek **and** Matthias Wählisch. "Gain more for less: the surprising benefits of QoS management in constrained NDN networks". **in**: *Proceedings of the 6th ACM Conference on Information-Centric Networking*. 2019, **pages** 141–152.

[41] Klaus Schneider, Cheng Yi, Beichuan Zhang **and** Lixia Zhang. "A practical congestion control scheme for named data networking". **in**: *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 2016, **pages** 21–30.

[42] Aytac Azgin, Ravishankar Ravindran **and** Guoqiang Wang. "pit/less: Stateless forwarding in content centric networks". **in**: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2016, **pages** 1–7.

[43] Sara Oueslati, James Roberts **and** Nada Sbihi. "Flow-aware traffic control for a content-centric network". **in**: *2012 Proceedings IEEE INFOCOM*. IEEE. 2012, **pages** 2417–2425.

Intentionally Left Blank

Publications Related to Thesis

Journal:

- Madhurima Buragohain and Sukumar Nandi. "Quality of Service provisioning in Named Data Networking via PIT entry reservation and PIT replacement policy". In: Computer Communications 155 (2020), pp. 166–183.

Conferences:

- Madhurima Buragohain and Sukumar Nandi. "LPECN: Leveraging PIT placement and explicit marking for congestion control in NDN". In: Proceedings of the 8th ACM Conference on Information-Centric Networking. 2021, pp. 20–29.
- Madhurima Buragohain, Chinmoy Jyoti Kathar, Chinmoy Kachari, Sunit Kumar Nandi, and Sukumar Nandi. "SCAN: Smart Collaborative Attack in Named Data Networking". In: 2020 IEEE 45th Conference on Local Computer Networks (LCN). IEEE. 2020, pp. 124– 133.
- Madhurima Buragohain, Prashant Gudipudi, Md Zaki Anwer, and Sukumar Nandi. "EQPR: enhancing QoS in named data networking using priority and RTT driven PIT replacement policy". In: ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE. 2019, pp. 1–7.

Book Chapter:

- Madhurima Buragohain and Sukumar Nandi. "Demystifying Security on NDN: A Survey of Existing Attacks and Open Research Challenges". In: The" Essence" of Network Security: An End-to-End Panorama. Springer, 2021, pp. 241–261.

Intentionally Left Blank

Brief Biography of the Author

Madhurima Buragohain born in Jorhat, Assam, India. She completed her Bachelors in *Computer Science and Engineering* from Jorhat Engineering College, India in 2013. She has received Gold Medal for securing first position in B.E. (Computer Science and Engineering). She received an M.Tech degree in *Information Technology* from Tezpur University, India in 2015 with a first-class (2nd rank). She is at present a regular research scholar at Indian Institute of Technology Guwahati, India under the guidance of **Prof. Sukumar Nandi**. Her area of Interest is Named Data Networking and Network security. She published her works in ACM ICN, IEEE ICC, IEEE LCN, COMSNETS conferences and Computer Communication (Elsevier).

Intentionally Left Blank

Index

Intentionally Left Blank

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati 781039, India