# Proactive Caching to support mobility in Named Data Networks

by

Hesham Farahat

A thesis submitted to the Graduate Program in the
Department of Electrical & Computer Engineering
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
September 2017

## Dedication

*To Mom, Dad, Rana, and Joana.*

# Abstract

Information-centric Networks (ICNs) offer a promising paradigm for the future Internet to cope with an ever increasing growth in data and shifts in access models. Different architectures of ICNs, including Named Data Networks (NDNs) are designed around content distribution, where data is the core entity in the network instead of hosts. Given the amount of forecast traffic by mobile users, supporting mobility in NDNs to maintain seamless operation is one of the main challenges yet to be resolved. Accordingly, attempts at handling mobility in NDNs in the literature are mostly studied under simplistic or special cases, and relied on content retransmission as a fallback. This is in addition to the lack of benchmarking tools to analyze and compare such schemes.

In this thesis, we investigate how predicting the future state of the network can enable seamless support of mobility in NDN. We propose a set of proactive benchmark solutions which exploit location and data traffic prediction to deliver the content of mobile users (both Consumers and Producers) under application delay constraints. Particularly, the network detects roaming users and caches their prospective content ahead of handover events while considering the maximum tolerable delay and network overheads. Unlike existing literature that focused solely on Consumer mobility, we also handle Producer mobility that impacts the content availability and Quality of

Service (QoS). Furthermore, we introduce a practical mobility management scheme that is resilient to prediction uncertainties using *stochastic* optimization. A *guided* heuristic search algorithm is also developed to provide real-time near-optimal caching decisions instead of commercial solvers that suffer from poor scalability.

All benchmark and heuristic schemes proposed in this thesis are evaluated using a comprehensive assessment framework, which is also used to assess the state-of-the-art NDN mobility support. Simulation results show that our proposed solutions maintain user's QoS during mobility events. We believe that such results drive incentives for deploying *proactive* mobility management in future NDN.

# Statement of Co-Authorship

The work presented in this thesis was accomplished under the supervision of Prof. Hossam Hassanein. The work of Chapter 3 was presented in [a]. Chapter 4 led to the work in [c]. The work in [d] and [f] is the result of Chapter 5. Whereas, the work of [b], [e] and [g] is presented in Chapter 6.

[a] H. Farahat and H. S. Hassanein. On the design and evaluation of producer mobility management schemes in named data networks. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems(MSWIM)*, pages 171–178, November 2015.

[b] H. Farahat and H. S. Hassanein. Optimal caching for producer mobility support in named data networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[c] H. Farahat and H. S. Hassanein. Supporting consumer mobility using proactive caching in named data networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2016.

[d] H. Farahat and H. S. Hassanein. Proactive caching for producer mobility management in named data networks. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 171–176, June 2017.

[e] H. Farahat, R. Atawia, and H. S. Hassanein. Robust proactive mobility management in named data networking under erroneous content prediction. [To be published in GlobeCom 2017], December 2017.

[f] H. Farahat and H. S. Hassanein. Decentralized proactive caching for producer mobility management in named data networks. Journal In preparation, 2017.

[g] H. Farahat, R. Atawia, and H. S. Hassanein. Robust predictive caching for producer mobility management in named data networking. Journal In preparation, 2017.

# Acknowledgments

In the name of God, the Most Gracious, the Most Merciful. Praises and thanks are due to God who bestowed upon us endless blessings, and the faculties of seeing, thinking, and learning. It is only with His guidance and sustenance that my endeavors and plans were seen through.

A special thanks for my supervisor Prof. Hossam Hassanein, your support since day one have led me to this place and your continuous guidance made a big difference in my research. I have learned a lot from you in the past 5 years. My experience in Queen's University would not be the same without your mentorship. Aside from the professional aspect, you were a true big brother for me and you made us feel home. It was of a great pleasure to get to meet your lovely family in Kingston. Wish you all the best and more success to come into your way.

I am also thankful to my thesis committee: Prof. John C. Cartledge, Prof. Saeed Gazor, Prof. Mohammad Zulkernine, and Prof. Halima Elbiaze. Your feedback during the last years helped to polish this work.

To Telecommunication Research Lab (TRL) members, it has been a pleasure to work with many of you. The weekly presentations were very informative and useful. I am grateful to Wenjie Li for the productive brainstorming about ICN, ns-3 and many technical topics, good luck in your PhD. Special thanks to Dr. Sharief Oteafy,

Dr. Sherin Abdelhamid and Dr. Mahmoud Qutqut for your academic support and advices throughout the past years. I would like to thank Mrs. Basia Palmer for her hard work to make TRL a great productive and efficient place.

I am grateful to Prof. Michael Greenspan for giving me the opportunity to teach in the ECE department. Special thanks to Ms. Debie Fraser for her dedication to help all graduate students. Many thanks to the GECE and Philip Oni, the social and sport events were very enjoyable.

I want to thank my Masters supervisors Prof. Maytham Safar and Prof. Khaled Mahdi. Your constant directions and influence led me to this place. To all my professors in Kuwait university, it has been a pleasure to meet and work with you. Dr. Mahmoud Abu Haswa, I am grateful for your engorgement and academic advices.

My precious parents, all the success in my life is because of you. There is no way I can pay back all the love, care, guidance and continuous prayers throughout my whole life. Thank you for being always behind my back and pushing me to achieve more in my life. To my beloved sisters Zeina and Nada, I am so grateful to have you both in my life. Your kids: Ali, Talia, Zidan, and Dalia bring joy to the family. Mohammed Ali and Mohammed Hakim, thanks for taking care of the family while I am away, wish you all the best in your lives.

To my partner in life Rana Ramadan, I would never have survived the last 5 years without you. Your sacrifice, love, and support made this work possible. My beloved daughter Joana, you brought all the happiness to our small family, May God bless you always. This work is dedicated for you. Thanks to my second family, Dr. Sami Ramadan, Dr. Eman Zaki, Sherif, Karim, Yusra and Mihav for all the support and your beloved kids Lily, Kaya and Tarek make our days colorful.

I am thankful to my friends in Kingston who made living away from home bearable. Thanks to Al-Saidis, Dr. Shadi Khalifa, Dr. Hatem Abu-zaid, and Dr. Fadi Badran. Special thanks to Dr. Sharief Oteafy and Dr. Layan Nahlawi for being a family to us in Kingston. I wish you all the luck in your new journey in Washington with your lovely daughter Reem. I am grateful to have a friend like Abdulla Abdelrahman, good luck in your life. To Anas Samir, I really enjoyed our daily discussions, all the best in your career.

To my true friend and brother Ramy Atawia, this work would not be complete without your support, guidance and encouragement. Thanks for everything. Your great sense of humor made our days enjoyable. You truly deserve the best, wishing you a bright future and may God grant you all the happiness with your partner Mariam El-Azab.

To my lifetime friends Jassim, Abdulhadi, Majed, Ali, Bassam, Ahmad, and Mohammed, thanks for always being there when I need you. I wish you the best of luck in your lives. I am lucky to have a friend like Ibrahim Sorkhoh, you are one of the best researchers I ever met. Good luck on your new journey.

Finally, I am very grateful for the funding provided by the Government of Ontario and Queen's University.

# List of Abbreviations

| | |
|---|---|
| ***OpCCMob*** | **O**ptimal **C**aching for **C**onsumer **Mob**ility. |
| ***OpProMob*** | **O**ptimal caching for **P**roducer **Mob**ility. |
| ***DCacheMob*** | **D**ecentrilized **Cach**ing for Producer **Mob**ility. |
| ***OpCacheMob*** | **Op**timal **Cach**ing for **P**roducer **Mob**ility delay-sensitive applications. |
| ***RCacheMob-Op*** | **R**obust **Cach**ing for Producer **Mob**ility - **Op**timal. |
| ***RCacheMob-RT*** | **R**obust **Cach**ing for Producer **Mob**ility - **R**eal-**T**ime. |
| **CCP** | Chance Constraint Programming. |
| **CDF** | Cumulative Density Function. |
| **CDN** | Content Distribution Network. |
| **DNS** | Domain Name System. |
| **DONA** | Data-Oriented Network Architecture. |
| **FIFO** | First In First Out. |
| **GA** | Gaussian Approximation. |
| **ICN** | Information-Centric Network. |

| | |
|---|---|
| **ILP** | Integer Linear Programming. |
| **ISP** | Internet Service Provider. |
| **LFU** | Least Frequently Used. |
| **LRU** | Least Recently Used. |
| **MSU** | Mobility Support Unit. |
| **NDN** | Named Data Networking. |
| **NetInf** | Network of Information. |
| **ns-3** | Network Simulator 3. |
| **P2P** | Peer-to-peer. |
| **PDF** | Probability Density Function. |
| **PMF** | Probability Mass Function. |
| **PoA** | Point of Attachment. |
| **PSIRP** | Publish-subscribe Internet Routing Paradigm. |
| **QoS** | Quality of Service. |
| **SA** | Scenario Approximation. |
| **SDN** | Software Defined Networks. |
| **SoCP** | Second Order Cone Programming. |
| **SUMO** | Simulation of Urban MObility. |
| **TCP** | Transmission Control Protocol. |
| **URI** | Uniform Resource Identifier. |

# List of Symbols

| | |
|---|---|
| $D$ | Set of data. |
| $d$ | Single data $d \in D$. |
| $u(d)$ | The Consumer of data $d$. |
| $u'(d)$ | The Producer of data $d$. |
| $\epsilon_d$ | The value of data $d$. |
| $R$ | Set of routers. |
| $r$ | Single router $r \in R$. |
| $c_r$ | The current cache capacity of router $r$. |
| $c_r^{max}$ | The maximum cache capacity of router $r$. |
| $c_r^{Sp}$ | The current proactive cache capacity of router $r$. |
| $c_r^{Spmax}$ | The maximum proactive cache capacity of router $r$. |
| $P_{r \to t}$ | The path from node $r$ to node $t$. |
| $\mathbb{P}_d^r$ | Number of path updates needed to cache data item $d$ on router $r$. |
| $\delta_r^d$ | Binary variable to cache data $d$ in router $r$. |
| $K_r$ | The total content value of router $r$. |
| $k_r^m$ | The content value of last $m$ items in router $r$. |
| $\rho_r^m$ | Binary variable to replace $m$ items from router $r$. |

| | |
|---|---|
| $\sigma$ | Threshold to bound the number of path updates. |
| $\alpha$ | Threshold to bound the number of cache replacement. |
| $\omega$ | Threshold to bound the loss in content value. |
| $\zeta$ | Threshold to bound the traffic generated by the Producer. |
| $\gamma$ | Threshold to bound the Consumer delay. |
| $\tilde{d}$ | Random variable to model the prediction uncertainty. |
| $S_d$ | The set of realizations of data $d$. |
| $s$ | A realization in set $S_d$. |
| $\pi^{d,s}$ | The probability that realization $s$ of data item $d$ will be requested. |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The evolution of social networking, mobile applications and media streaming caused a shift in how the Internet operates. With an expanding use of the Internet, Cisco's Visual Networking Index (VNI) [25] projects a threefold increase in global traffic—to reach 132.8 Exabytes per month—by 2021, compared to 2016. While the Internet was built on a host-to-host model, current usage trends are exhausting network resources in maintaining scalable operation. Consistent attempts at patching up Internet operation to cater for increasing content are bound to fail under the projected increases in data traffic [102]. This includes attempts at supporting Content Distribution Networks (CDNs) and Peer-to-peer (P2P) overlays on the IP network.

Recently, research efforts were directed at developing a paradigm for the future Internet based on *content* rather than *hosts*. A promising paradigm, namely Information-Centric Network (ICN) [18], was presented to address growing challenges with content oriented communication. While considering the strengths and weaknesses of the current Internet design, ICNs are being developed in concerted efforts,

and several potential architectures have been proposed. The commonality between all ICN designs is that content is identified by unique names and can be cached anywhere in the network. While most of the attention has been directed at addressing challenges such as: routing, naming, and caching, other important challenges remain seldom tackled [96].

A core property of any future Internet architecture is supporting mobility as a networking primitive. This is exacerbated with a projected increase in mobile entities amounting to over 50% of all devices and connections by 2021 [26]. Supporting seamless operation where traffic receivers, *Consumers*, and content providers, *Producers*, can move in the network without service interruptions is core to ICNs. This challenge is gaining attention as researchers are attempting to incorporate mobility support in ICN architectures. In the current Internet, the challenge of delivering traffic from or to mobile devices is in how to find the moving hosts in the network. Whereas in ICNs, the challenge is in how to find and track the data [103].

Named Data Networking (NDN) is one of the pioneering ICN architectures, that was proposed by PARC [103]. NDN is assumed to support mobility intrinsically by retransmissions. However, this has shown to be impractical and sub-optimal due to the excess delay and wasted network bandwidth. While recent research efforts have addressed mobility challenges in ICNs generally, and NDNs specifically, there remain significant challenges in analyzing the impact of these solutions on the resulting Quality of Service (QoS) and latency measures.

In this thesis, we investigate how *predictions* can be used to provide proactive mobility solutions for ICNs. By knowing when users move to another network and what data is requested, proactive decisions can be taken to cache the future content to

avoid delays and retransmissions. For instance, if a user is broadcasting a live stream on YouTube while roaming between networks, part of the content can be cached in the network *before moving* such that users watching that broadcast do not notice the downtime.

In essence, the motivations behind this research are:

1. The potential of NDN to incrementally replace the current Internet design.
2. The projected growth of mobile traffic generated/requested by devices roaming in the network.
3. The practicality of predicting users' locations and possible network handovers given accurate positioning using navigation-enabled devices.
4. The availability of data predictors that can estimate data requests using history information.

## 1.2 Research Statement and Thesis Contributions

In this thesis, we investigate the problem of mobility in NDN. Mainly, we try to answer the following research questions:

1. What is the effect of mobility on the performance of NDN?
2. Does proactively caching data using predictions improve the performance of mobility management scheme?
3. How resilient are proactive mobility schemes to prediction errors?

4. Can we devise a robust proactive mobility scheme to handle erroneous prediction?

First, we study the impact of mobility on NDNs and assess the performance of current proposals of mobility support. This is achieved by developing a comprehensive assessment framework. Second, we propose optimal solutions to be used as benchmarks for both Consumer and Producer mobility problems using perfect knowledge of the future. Third, sensitivity analysis is done to evaluate the resilience of the proactive schemes by introducing typical errors to predicted locations and data requests. Finally, a robust real-time solution is proposed to support mobility.

The contributions of this thesis are summarized as follows:

1. A comprehensive modular assessment framework is developed within Network Simulator 3 (ns-3) [75] to evaluate the performance of existing and proposed mobility management schemes under various network topologies. Heterogeneous mobility, access and traffic profiles of Producers and Consumers are adopted in the evaluation; to be released for NDN research. Such a framework allows detailed analysis of proposed mobility management schemes in NDN, with insights on design factors that yield to seamless mobility. This addresses the first research question.

2. We propose an optimal proactive caching benchmark for the Consumer mobility problem, named *OpCCMob*. First, we formulate the problem as content placement based on the forecast Consumers' locations. Then, an optimal solution is obtained using Gurobi optimization solver [47]. Moreover, we unleash the performance gains due to predictions compared to popular state-of-the-art mobility management schemes. The Consumer mobility part of research questions

2 and 3 is tackled here.

3. We propose a proactive caching optimization framework that exploits users' positions and traffic prediction to support seamless Producer mobility in NDN with minimal overhead required. We design an optimal benchmark, *OpProMob*, that can be used as a baseline for other mobility schemes. Furthermore, we propose a polynomial time algorithm, *DCacheMob*, that provides a decentralized near-optimal solution in real-time by greedily finding the optimal placement of each predicted *interest* in network caches. The Producer mobility part of research questions 2 and 3 is tackled here.

4. All the aforementioned optimal schemes are evaluated under accurate prediction of future information, which may not be feasible in practice. Hence, we present a robust proactive cache optimization framework for Producer mobility with delay-sensitive applications. First, we propose an optimal benchmark, *RCacheMob-Op*, that is resilient to prediction errors by using Chance Constraint Programming (CCP). Second, a practical real-time algorithm, *RCacheMob-RT* is developed to provide near-optimal solutions to the proactive caching problem at hand. This addresses the last research question.

## 1.3 Thesis Outline

In this chapter we have stated the research problem, highlighted the motivations and discussed our major contributions. The remainder of this thesis is organized in six chapters as follows.

Chapter 2 gives an overview on ICNs and the problem of mobility. Then, a survey of research efforts targeting mobility support in NDN is presented.

Chapter 3 presents the proposed assessment framework, which is used to evaluate mobility solutions in NDN. The experimental design, performance metrics and factors used in later chapters are then discussed.

In Chapter 4, we propose an optimal benchmark solution to handle Consumer mobility in NDN. Assuming perfect knowledge of the future, the problem is formulated as an Integer Linear programming model which can be solved by commercial solvers to measure the gap of performance in any Consumer mobility scheme.

Chapters 5 and 6 tackle the Producer mobility problem. First, in Chapter 5, we design a benchmark that finds the optimal placement of content with limited overhead. Then, we develop a decentralized heuristic scheme that can find a low complexity near-optimal solution in real-time.

In Chapter 6, we provide an optimal scheme that solves Producer mobility for delay-sensitive applications. Then, a robust benchmark scheme is proposed which uses stochastic optimization to find the optimal content placement without violating the application delay requirement. Finally, we present a robust real-time heuristic that finds a near-optimal solution at polynomial complexity.

Chapter 7 presents a summary of this research, concluding remarks and possible future directions.

# Chapter 2

# Background and Overview

This chapter is divided into two parts. The first part motivates the need for ICNs in general, and provides a brief background on the different proposals for the future Internet.

The second part is devoted to mobility support in ICNs. The problem of users mobility and its impact on network performance is explained in details. This is followed by reviewing and analyzing proposed solutions in the literature.

## 2.1 The Current Internet Usage

The amount of information transfered over the Internet is increasing every day. A recent study from Cisco [27] shows that in 2016, the amount of global IP traffic was 1.2 Zettabyte[1]. Moreover, Cisco has predicted, in [25], that the amount of traffic will increase threefold by 2021, forming a 100-fold increase from 2005 to 2021.

We think that the reasons behind this tremendous amount of data are:

1. *The evolution of users*:

[1] 1 Zettabyte = 1 billion Terabytes

The number of Internet users is increasing every year. Originally, the Internet's users were mainly academics, researchers and organizations' employees. But as of today, 47% of the world population (3.4 billion users) are connected to the Internet[2] [52].

2. *The evolution of devices*:

   It is predicted that in 2021, the number of devices connected to the Internet will be nearly three times the global population. Moreover, there will be on average 3.5 devices per person in 2021 compared to 2.3 devices in 2016. Quantity of devices is not the only game-changer, the technology is evolving as well. In the early days of the Internet, devices were mostly servers and mainframes. Later, the Personal Computers and then Laptops were introduced, allowing regular users to connect to the Internet from anywhere. The evolution then witnessed connecting powerful cell phones to the Internet which created the booming smart phone market. Currently, new concepts are being commercially introduced and used by many users, such as the wearable devices and Internet-connected vehicles.

3. *The evolution of content*:

   The type of content in the Internet has been changing, since the start of the Web, due to new technologies being created and used in different applications of the Internet. The Web, for example, started with texts in web pages, but now full high definition movies can be streamed to regular users [3]. Not only the

[2] 81% of developed world's population are connected to the Internet, whereas just 40% the developing world's are using the Internet

[3] 78% of the world's mobile data traffic will be video by 2021 [26]

type is changing, but also the quality. With all the new technologies in photos and movies, content size is increasing exponentially.

Web 2.0 is another example which is a collection of technologies used to add dynamicity to the earlier static web sites [46]. Moreover, the use of Web 2.0 has allowed users to share and collaborate content on the World Wide Web (e.g., social networks, blogs, wikis, etc.) which is another huge source of content that is being produced every second [71, 72, 85, 86]. There is also the data produced and consumed by machines (Machine-to-Machine communications) which is an extra load on the network and should be handled efficiently.

## 2.2 The Current Internet Design

The communication model of the current Internet was created in the 1960s. The core model, influenced by the telephone system, is based on host-to-host communications which has not changed since then [53]. However, the current usage of the Internet is changing. In particular, current applications of the Internet are content-based, where the main concern for users is the *content* not the *host* providing the *content*. Running throughput hungry and delay sensitive content-based applications on a host-based network needs extra features that are not supported by the current core architecture. Consequently, patching up the Internet with overlay protocols such as CDN and P2P is the current solution to cope with the high demand.

CDNs solves the problem of availability of content in the Web. It aims to increase the performance of content delivery by bringing it closer to the users [76, 89]. In particular, CDN replicates the content on different sites, called *surrogates*, which are installed in independent networks. A request for a certain content is then redirected

to one of the surrogate servers selected based on network congestion and servers load. Usually, CDN services are commercially offered for content providers by dedicated companies which makes it not affordable by every publisher.

P2P is a network of peers that contribute resources to cooperatively provide services for each other without central control [76]. This enables distributed scalable content delivery that does not need operator support. Specifically, it creates a network abstraction that is considered as an overlay to the real network. P2P applications such as file sharing, streaming and Video-on-demand are built on top of this abstraction.

Both overlays protocols are implemented at the application layer without changing the Internet core. However, this adds complexity to the architecture and are bound to fail with the prospective traffic [100]. For example, CDNs typically use network-unaware mechanisms, which lead to inefficient utilization of network resources. Additionally, P2P design has many challenges in mobility and security. Hence, researchers are designing a new *Internet core* to be built specially for *content*.

## 2.3 The Future Internet

By considering the strengths and weaknesses of the current Internet, a new content-based design is being investigated. In particular, the future Internet must take into consideration the change in users' perspective and focus on data, the "*what*", rather than hosts, the "*where*". Specifically, the new communication model should be a content-based which considers *data* as the main block. Any new proposal for the future Internet must provide the following features [30, 53]:

1. *Efficient content distribution*:

The new network model must adapt content distribution methods in its core (i.e., without patching).

2. *Location independent naming:*

   Addressing hosts should not be a problem to higher levels. It should avoid content to hosts coupling, since this complicates the design and operation of upper layers.

3. *Availability:*

   Fast and reliable data access.

4. *Content security:*

   Provide authenticated and encrypted content by applying security on the content, not the path that may include untrusted elements.

5. *Mobility:*

   The new paradigm has to handle the movements of mobile users. Moreover, it should not restrict content mobility (i.e., movement of content providers).

6. *Incremental roll out:*

   Moving towards new communication model has to be seamless (i.e., minimum effect on users and implementation cost), thus an incremental approach should be taken [40].

The new paradigm of ICN considers these features as core design requirements to build a content based network. There are several promising proposals for an ICN design:

1. Named Data Networking (NDN) [53].
2. Data-Oriented Network architecture (DONA) [58].
3. Network of Information (NetInf) [30].
4. Publish-subscribe Internet Routing Paradigm (PSIRP) [61].
5. Translating Relaying Internet Architecture integrating Active Directories (TRIAD) [24].
6. 4WARD [74].
7. Scalable and Adaptive Internet Solutions (SAIL) [41].
8. Content mediator architecture for content-aware networks (COMET) [43].
9. MultiCache [55].

The proposed architectures have similarities and differences which are introduced in Section 2.4 and the first 4 architectures are briefly explained.

## 2.4 Information-Centric Networking

### 2.4.1 Basic Design and Common Characteristics

The ICN paradigm proposes a new communication model with new characteristics which are explained here.

### Content naming

Packets are content carriers in the Internet, they travel the network to transport data between two hosts identified by IPs only. Moreover, packets are routed using the receiver's IP which is attached to the packet. However, to design a network with content as the main block, the relation between hosts and information should be *decoupled*. Specifically, content should be named such that packets are routed using the name not the host's IP. Ideally, the naming scheme must provide the following [18]:

- Uniqueness: content should be uniquely identified.
- Persistence: guarantee the validity of the content name.
- Scalability: the name-space scales-up with the amount of content with no limitation.

At that end, two major naming schemes are being used in the ICN proposals:

1. Hierarchal naming:

   This scheme is similar to Internet's Uniform Resource Identifier (URI) concept, where the name is structured hierarchically to easily locate the content. The name typically comprises the content provider, content's name and version or segment. Moreover, hierarchal names are readable.

2. Flat naming:

   This scheme uses self certifiable names [45], the name is random bits created from the content itself not the location (e.g., hash of the content). However, the name in this case is not readable.

### Requester-driven model

Unlike the current design of the Internet, where communications are initiated by the source, communications in ICNs start at the receiver. Hence, the packet is replaced by two primitives, one is to request data and the other is to carry the data itself. This concept is similar to publish/subscribe models, where *Subscribe* is used to request data and *Publish* packets are used to advertise data. The names of the primitives differ between architectures. For example, in NDN [53] the primitives are Register/Interest, in DONA [58] Register/Find and in PSIRP [61] Publish/Subscribe.

### Routing

Since the new packets (Request and Data) are not IP-based (i.e., no IP addresses attached to the packet), the new paradigm would have a new routing methodology. The routing has two phases:

1. Named-based Routing: routing the request to the data source. (i.e., request packet routing)

2. Routing back the data from the source (or stored copy) to the requester (i.e., Data packet routing)

Different routing techniques are used in each ICN architecture which will be explained later in this chapter.

### Content Caching

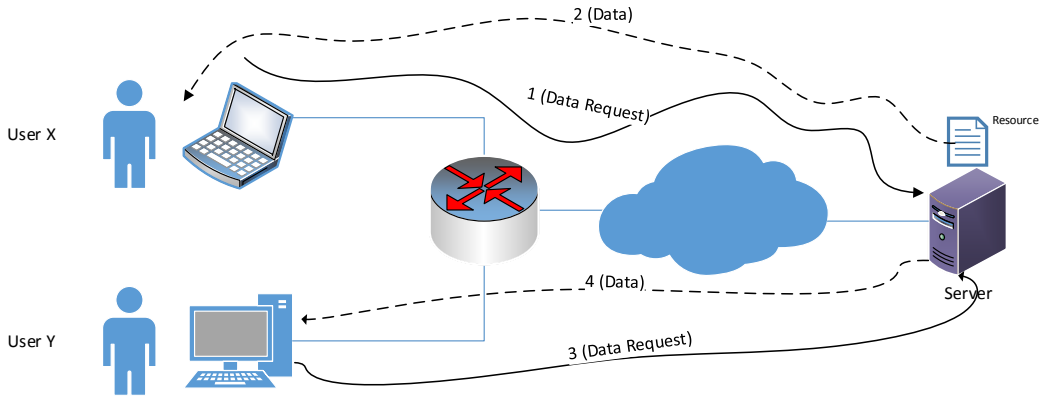
With the daily increase in the amount of requested content, the concept of caching became of paramount importance [51]. Content caching refers to storing data, for a

definite time, in places other than the source (i.e., usually closer to end-users) such that future requests of the same data can be satisfied much faster [67,100]. Moreover, adopting an optimized caching policy, servers would not waste resources answering the same request from multiple users. Additionally, the network infrastructure would save resources required for data forwarding. Consider the scenario where thousands of users located in the same geographical area are interested in streaming the same video online. At the beginning, each user initiates a session with the web server providing the video. Then, the server will have to send the content in multiple packets to each connected user, passing through the same intermediate routers. If caching is possible, the server will have to send the packets only once, and Internet Service Provider (ISP) routers would cache the packets in order to send them directly to users. Figure 2.1 shows how content caching at routers can save network resources.

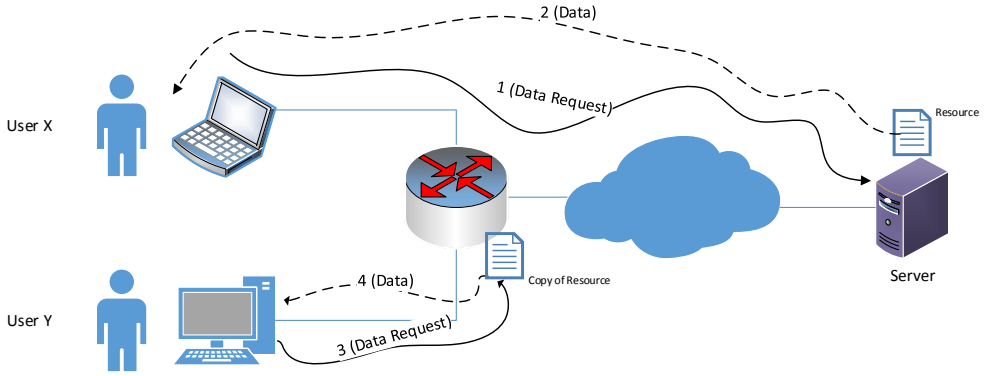
Unlike CDNs, where caching is allowed in specific points in the network, ICN caching can be done at any node in the network. All nodes, whether it is an end-user device or an ISP router, will be able to cache content passing through it. *What* to cache and for *how long* are decisions that affect the performance of the network. For instance, content's popularity, size and version are some factors that the network designer should consider, to optimize the cache performance [51, 84].

ICN caching schemes can be classified into two classes, on-Path and off-path caching [51]. In on-path caching schemes, content is cached at nodes along the data forwarding paths. Furthermore, on-path caching can be probabilistic [78], graph-based [21] or popularity-based [79].

On the other hand, off-path caching schemes replicate data in the network, to increase availability, regardless of the forwarding paths. Central entities (e.g., ISP)




(a) No caching in routers




(b) With caching in routers

Figure 2.1: Example of content caching in Information-Centric Networks.

are responsible for deriving a caching strategy that strikes a balance between content availability and caching overhead [1,66]. This type of caching is used to support other functions in ICNs as explained later in this thesis.

### Security

The security model used in the current network is based on securing the path (channel) of content [5,53]. This model requires the client to trust the server and all intermediate nodes on the path. In ICNs, the security schemes are applied to the content itself

instead of the path [45]. Hence, the content can be provided by nodes other than the data provider. Protecting the content provides name-data integrity and authenticity anywhere in the network.

### 2.4.2 ICN Architectures

In this section, four of the major ICN Architectures, NDN, DONA, PSIRP and NetInf, are briefly explained. These ICN architectures are more recent than others and are being actively developed [5, 45, 100]. Concepts like Naming, routing, caching and security are investigated for each architecture.

#### Named Data Networking

Named Data Networking (NDN) is one of the main ICN architectures that is widely used as a base architecture for researchers in this field. The architecture was originally proposed by PARC as a project named Content-Centric Networking (CCN). Now NDN is an NSF-funded Future Internet Architecture project [53, 100]. The vision of the proposal is to reshape the Internet protocol stack by making *named data* the new thin waist instead of the IP protocol.

There are two types of packets in NDN, *Interest* and *Data* packets. The former is sent by users (Consumers) to request content, whereas the latter is generated by content providers (Producers) to match the *interest* packet. Moreover, hierarchical naming is used in this system which makes the routing process similar to the IP network.

NDN uses three main data structures implemented in each node [5, 53, 100]:

1. Forwarding Information Base (FIB):

A routing table which is used to forward *interest* packets to the next router. It is very similar to the routing tables in the IP network, however content names are used instead of hosts' IPs.

2. Pending Interest Table (PIT):

   Structure to keep track of previously forwarded *interests* and the requester of those *interests*, such that when the *data* is available it can be sent back to the requester.

3. Content Store (CS):

   The cache where the *data* can be stored. If the *data* is cached in the CS, the node can reply with the *data* without forwarding the *interest* (i.e., satisfy the *interest*).

Using the three data structures in all NDN nodes enables *interest* and *data* forwarding. In particular, the *interest* packet is handled by looking up the content name in the Content Store, to check if the *data* is cached. If there is no entry in the CS, the PIT is checked for older *interest* that may have requested the same *data* before. In this case, there is no need to send another duplicate, and the *interest* will be dropped. Finally, if the *interest* is not found in the PIT, the FIB is checked to find where to forward the *interest* and a record of this *interest* will be added to the PIT. It has to be noted that, if the *interest* cannot be found in all the *data* structures, then the *interest* is discarded, since the node does not have the *data* and does not know how to forward it [53]. Figure 2.2 shows the flow chart of this process.

The *data* packet is forwarded back to the Consumer using the same path used for the *interest*. This is possible by using the *breadcrumbs* that were added to the

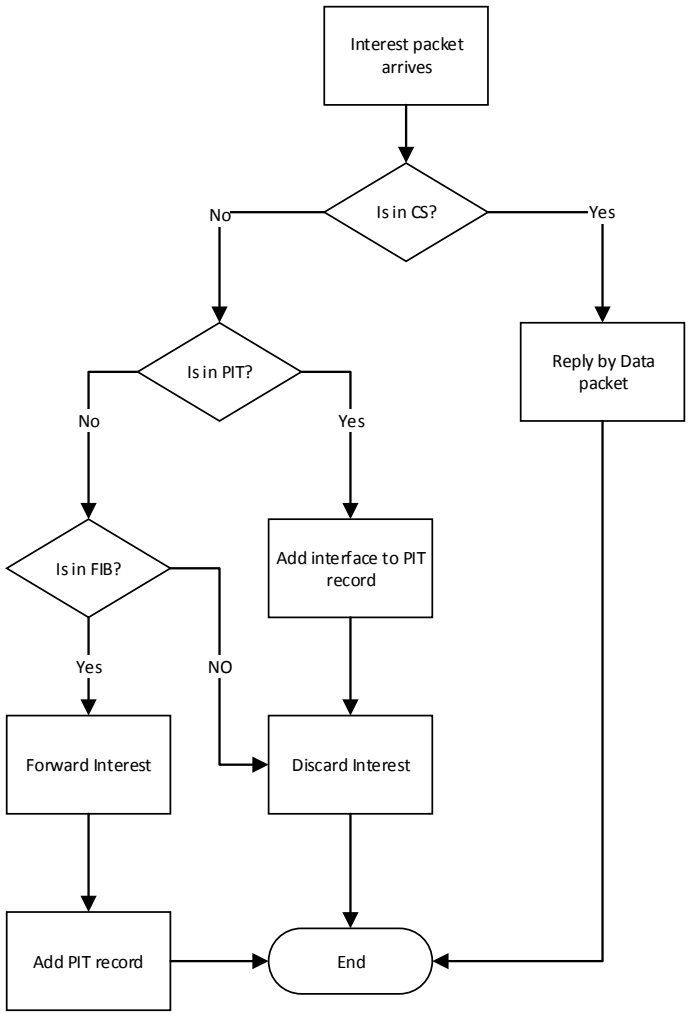



Figure 2.2: Flowchart of *interest* handling in NDN.

PIT every time it passes through a new router. Once the *data* packet arrives to a node, a decision will be made whether to store the *data* in the Content store or not. This is an example of on-path caching, i.e., intermediate nodes can cache the *data* while it is traveling to the requester. Then, the *data* will be forwarded to all interfaces in matched entries of the PIT table [53]. Figure 2.3 shows an example of *interest* and *data* packets flow in the network. NDN security mechanism is applied

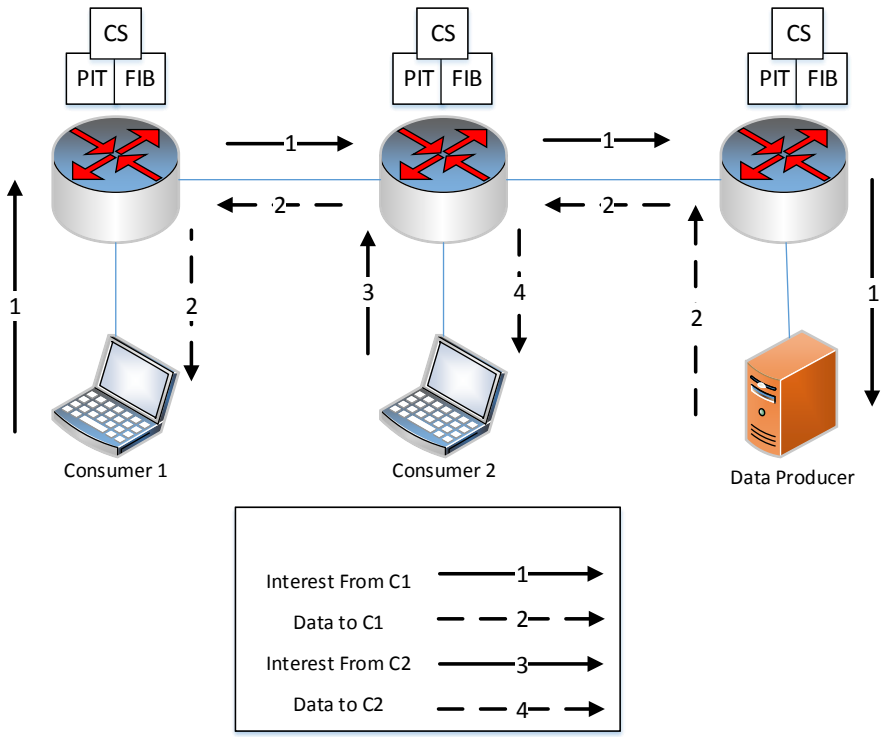



Figure 2.3: *Interest* and *data* forwarding in NDN, Consumers 1 and 2 send *interests* for the same *data* from the provider.

to the content [18]. The content can be authenticated with digital signatures and Producer's public key [53].

### Data-Oriented Network Architecture

Data-Oriented Network Architecture (DONA) [58] is one of the first ICN architectures which gives a new design to content naming and name resolution in the Internet [58]. The main idea of DONA is to add a service similar to the Domain Name System (DNS) but for content instead of hosts [58]. In DONA, the contents have flat names in P:L format, where $P$ is the hash of the publisher public key and $L$ is an arbitrary content label [18]. This type of name is self-certifiable; using the name and the content

the network can authenticate the data received.

Name resolution (name routing) in DONA is done using a new class of network entities known as Resolution Handlers (*RH*) [5]. Moreover, the two basic primitives needed to accomplish the name resolution are *FIND(P:L)* and *REGISTER(P:L)*. The requester creates *FIND* packet to ask for content, *RH* routes these packets towards nearby copy. Information of locations of nearby sources are known from the *REGISTER* packets, which were sent by any authorized provider of the content. Resolution handlers should be in a hierarchical setup (i.e., levels/tiers), each domain has one local *RH*, which is connected to another *RH* in different level. Each *RH* has a table that keeps the information of the next *RH* for this specific content. An example of the complete process is shown in Figure 2.4. *RH* is responsible for routing the *FIND* and *REGISTER* packets [5, 58]. With regards to content delivery, DONA does not require changes to the current IP network design. Extensions to DONA can add more functionality to the architecture. For example, caching is possible on *RHs* by changing the requester address in FIND packets to its address such that data passes through the *RH* first.

One of the advantages of DONA is that it works with the IP network [5]. Specifically, it should work as a new layer between the IP and Transports layers in the Open Systems Interconnection (OSI) model. For instance, *FIND* packets will be addressed with IPs and padded with transport layer headers. Moreover, the transport layer should change from host addresses to content names, which will simplify applications design.

In DONA, content can be authenticated by just using the name of the content P:L, the public key of the publisher and the signature of the data [100].

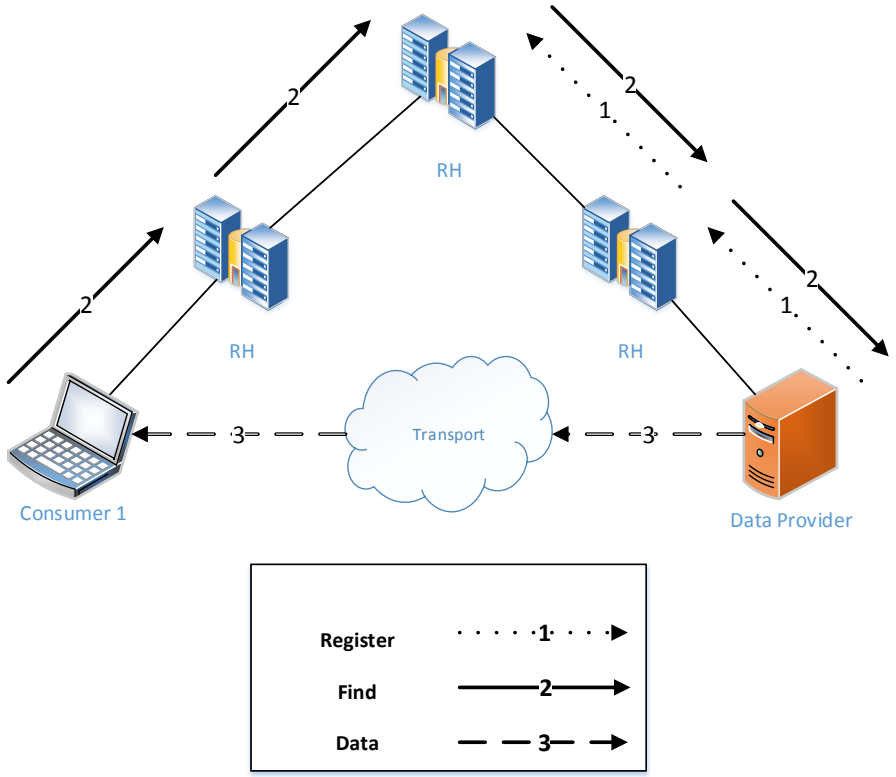



Figure 2.4: Packets flow in DONA.

### Publish-subscribe Internet Routing Paradigm

Publish-subscribe Internet Routing Paradigm (PSIRP) replaces the IP protocol stack with a publish-subscribe protocol [61, 100]. The content is flat-named with two identifiers, scope ID (a group of information) and rendezvous ID (information ID). In details, the architecture consists of four functions:

1. Rendezvous System: is responsible for matching subscription to publications.

2. Topology Function (using Topology Nodes): manages the physical network topology.

3. Routing Function (using Branching Nodes): builds and updates routing trees for each publication and cache content.

4. Forwarding Function (using Forwarding Nodes): actual data delivered to subscribers.

Name resolution and data routing can be summarized as follows:

1. Publish Information: publishers (content provider) publishes the information to the Rendezvous network.

2. Subscribe information: content consumer subscribes for information by sending the data name to the Rendezvous network.

3. Publish-subscribe match: The Rendezvous network matches the publish information to the subscribe request.

4. Using topological information and performance metrics from the topology function, the branching nodes will create Forward Identifier which will be sent to the publisher. The branch nodes may contain a cached copy and will forward the data to the subscriber.

5. Publishers uses the Forward Identifier $F_{ID}$, to forward the data to the Consumer. The data forwarding is done using the $F_{ID}$s and the link IDs to determine the forwarding nodes.

The flow chart of execution is shown in Figure 2.5. As mentioned earlier, branch nodes are responsible for caching. However, it may not be very effective since the name path may differ from the data path. Other methods of caching are proposed in [100].

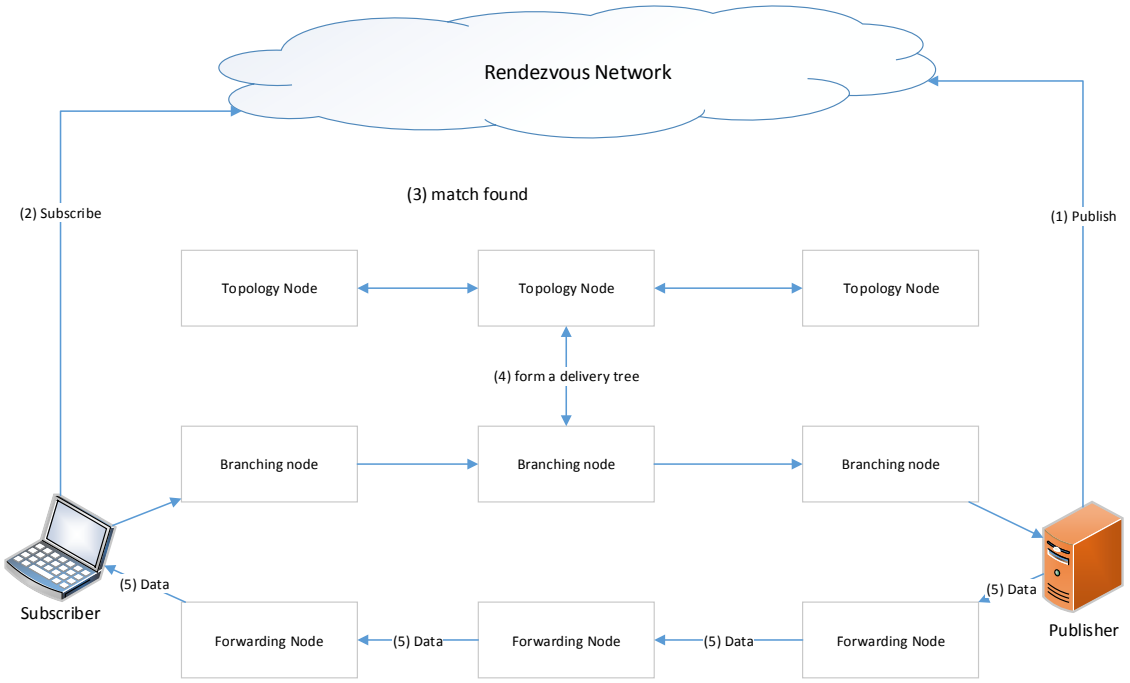



Figure 2.5: Packets flow in PSIRP.

PSIRP uses Packet Level Authentication (PLA) which provides integrity, authenticity and accountability on the network layer [61, 100].

### Network of Information

Network of Information (NetInf) uses named objects (content) instead of hosts [30]. It employs flat naming model, hence name resolution service should be used. To be able to publish data, data providers register the location and the name of the data in the Name Resolution Server (NRS).

NetInf supports both techniques name resolution (using NRSs) and name-based routing to find the content [30]. In particular, the requester sends *GET* packet asking for the data, the packet will travel on hop-by-hop basis until it finds the data (or a

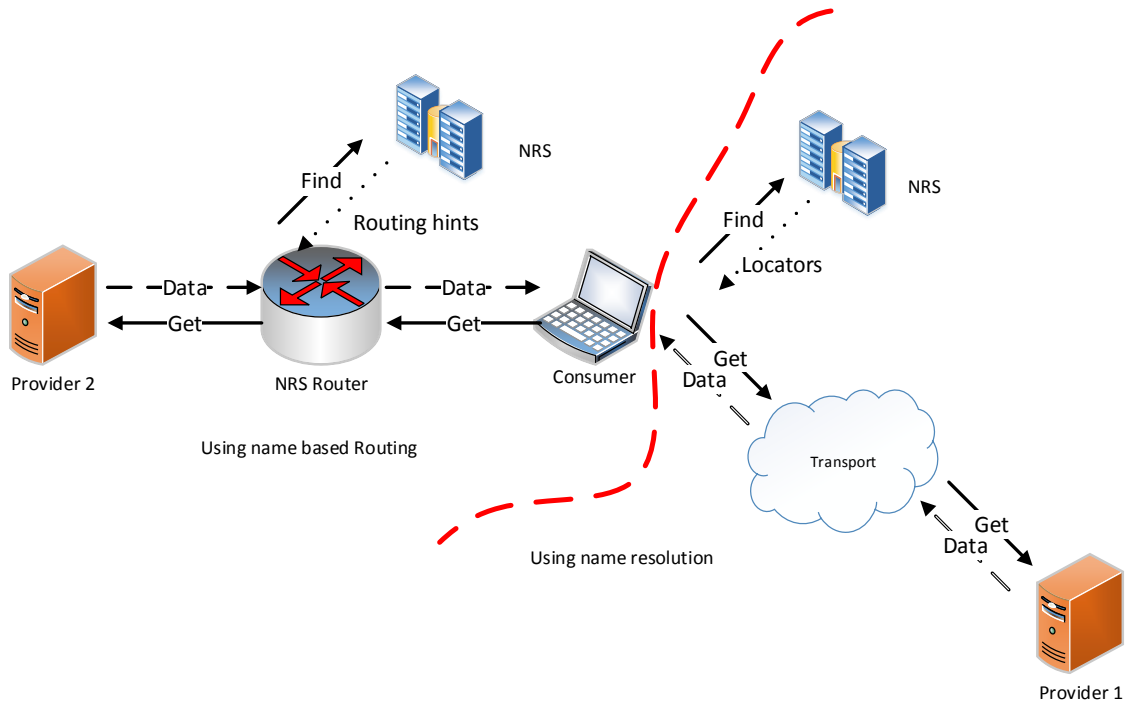



Figure 2.6: Packets flow in NetInf. Two modes are shown here: Name resolution and named routing.

cached copy). If the next hop information in the router is missing, the router will ask the NRS for the location of the data source [5]. Figure 2.6 shows the two modes and the messages flow in the network

Caching in NetInf can be both on-path and off-path. NetInf routers have built-in caches, which enables caching content on the request/data path [30]. Off-path caching is controlled by the NRSs, which will choose specific locations in the network to cache content. In particular, when a request arrives to the NRS, it may direct the requester to one of the off-path caches.

The names in NetInf contains security information to provide name-data integrity and other advanced security features [30].

Table 2.1: Comparison of the main four architectures.

| Arch | Naming | Request Routing | Content Routing | Caching |
|---|---|---|---|---|
| **NDN** | Hierarchical | Named-based Routing | Reverse path | Cache everywhere |
| **DONA** | Flat | Name Resolution | IP | Cache on RH |
| **PSIRP** | Flat | Name Resolution | Source routing using Bloom filter | Cache on BN |
| **NetInf** | Flat | Hybrid | Reverse path or IP | Cache everywhere |

### 2.4.3 Discussion

The architectures have differences in the implementation of ICN's functionalities [100]. In Table 2.1, the functionalities of the 4 architectures are summarized [5, 18, 100]. The content naming scheme is a fundamental design choice which impacts other functionalities such as routing, security and mobility. Some architectures propose a complete new design such as PSRIP and NDN, whereas others use IP for content routing as in DONA and NetInf.

## 2.5 Mobility Support in ICN

Content generated or consumed by mobile devices has a big share of the global traffic. The demand of mobile users is increasing every day where it is estimated to reach 20% by 2021 which is a sevenfold increase from 2016's statistics [26]. Mobility of nodes in the Internet started with laptops, which were designed to be mobile and connected with no wires to the Internet. Cell phones are another example of mobile nodes which were designed for voice communication. However, the data generated by such devices is a large percentage of the total mobile traffic due to the enabling

technologies such as LTE and smart devices.

Given that the coverage region of mobile cells and access points are limited (varies from meters to kilometers), and users are mainly moving (in vehicles or walking), devices may have to reconnect to a new network closer to it. This process is called handover or handoff. Furthermore, there are two types of handovers, vertical handover where devices need to switch technologies (e.g., WiFi to 3G) or horizontal handover where devices switch between different access points in the same network.

To meet QoS requirements, the handover should be done in a seamless operation. In other words, the applications running during a mobility event should face minimal interruptions. At the same time, the network should not be overloaded with extra traffic due to such events. These requirements are critical for real-time and time sensitive applications such as video conferences and live TV streams. Consequently, protocols and solutions to handle mobility events are inevitable for successful network operations.

In the following subsection, the current solution for mobility support in the Internet is discussed. Next, mobility solutions in ICN architectures are discussed.

### 2.5.1 Current Mobility Support Protocols in the Internet

The current Internet model does not support mobility by default, therefore mobility protocols were designed and used with the current designs. Internet mobility schemes can be classified into two categories [107], namely routing-based and mapping-based. The first approach supports mobility through dynamic routing which does not need explicit mapping to find the location of a node. Examples of this scheme are Columbia [50] and HAWAII [80]. This approach is simple but is not scalable, since all nodes

should be notified of every other node's movements. The second approach is based on mapping the node's identifier to the current location. In this case when a node moves, it should notify the mapping server, not all nodes in the network. Mobile IP [95] and VIP [94] are examples of this class.

#### Mobile IP

The Internet Engineering Task Force (IETF) has developed Mobile IPv4 and IPv6 standards in 2002 and 2004, respectively. The basic idea behind the protocol is that each domain has a *Home agent* (*HA*) which provides the *Home address* (*HoA*) for its local nodes. This address is a stable identifier that other nodes will use to reach this specific node. When the node moves to a new domain, it obtains a *Care-of address* (*CoA*) from the current router it is connected to. Moreover, the new address should be sent to the *HA* to update its mapping table. If a fixed node *CN* needs to send packets to a mobile node *MN*, it will be directed to *HoA*, then the home agent will encapsulate packets to the *CoA* to reach the mobile node in the new location. Triangular routing is a problem of such approach, where different routing is used for different directions. In particular, packets are going from *CN* to *HA* and then forwarded to *MN*, whereas the reply goes back from *MN* to *CN* directly.

### 2.5.2 Mobility Support in ICN Architectures

As stated in Section 2.4, there are two types of nodes in ICN: Consumers (*data* requesters) and Producers (*data* creators). Producers are used to be stationary since data are usually hosted in servers at fixed locations. However, with the increase use of social networks, regular users are now "pronsumers" where they produce and consume

data. Therefore, any mobility solution should handle both Consumers and Producer mobility events. In the next subsections, we will explain how the four chosen ICN architectures handle both mobility types.

### Named Data Networking

NDN uses late binding in which content is matched to a location later in the routing process [96]. It was claimed by NDN designers that this feature simplifies mobility handling in NDN and it is intrinsically supported with no need for extra logic. For instance, Consumer mobility can be simply solved by retransmitting *interests* after the handover. Consider the following scenario:

1. Consumer $X$ sends an *interest* to Producer $Y$ requesting a specific content.
2. *Data* from $Y$ will be forwarded to $X$.
3. Before *data* reaches $X$, it moves to a new network (connects to a different router).
4. *Data* arrives at old location of $X$ but will be cached in intermediate routers.
5. $X$ will wait for a predetermined period of time and then resends the *interest* again.
6. *Data* will be forwarded again, but *hopefully* from a shorter path using a cached copy.

Reissuing *interests* after timeout periods may be a practical solution for regular applications. However, it is impractical for delay-sensitive applications since the extra time caused by the handover can exceed the limits of QoS. Moreover, the network

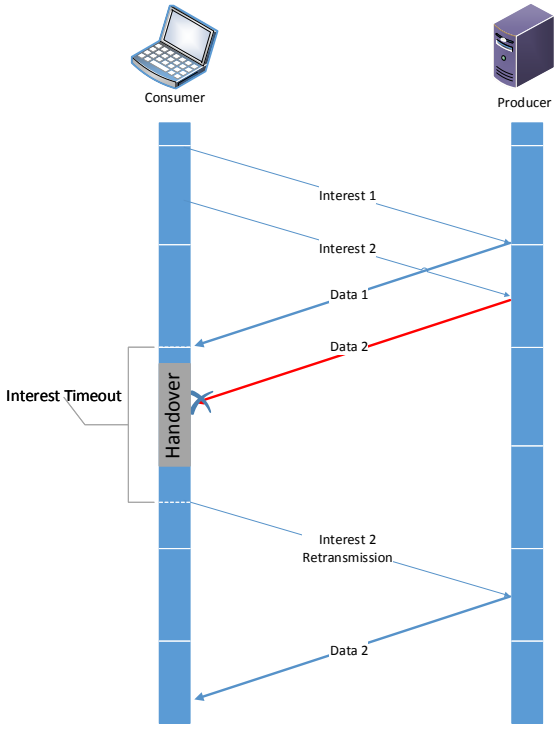



Figure 2.7: Timeline of Consumer mobility in NDN.

needs to handle forwarding of duplicate *interests* which is an overhead on the network. Figure 2.7 shows the timeline of events during Consumer mobility event.

In the case of Producer mobility, the *data* provider changes its location and therefore routing tables (i.e., FIB) need to be updated with the new location. The time taken to update the routing state of all routers in the network is called *routing convergence delay*. Since the Producer is not reachable during this period, all interests will be dropped and retransmissions are needed. This simple solution is not scalable, since the convergence time is long for large scale networks. Figure 2.8 shows the timeline of events during Producer mobility event. Additionally, the authors in [16, 69] have conducted simulations to study the capability of NDN in handling simple mobility events. The results have shown that NDN with its typical architecture fails to provide

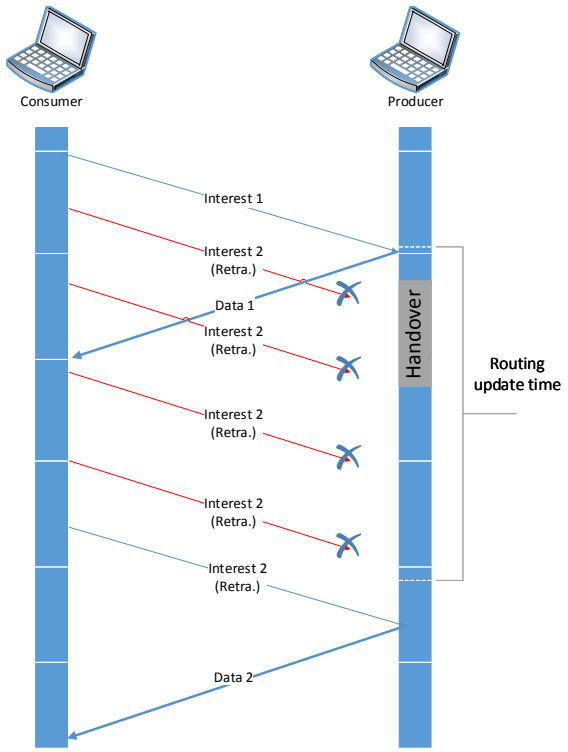



Figure 2.8: Timeline of Producer mobility in NDN.

seamless mobility; therefore the use of an effective mobility management scheme is necessary.

### Data-oriented Network Architecture

In contrast to NDN, DONA uses early binding by deploying Resolution Handlers (see Section 2.4.2 ), where the content is mapped to location before routing. Consumer mobility is handled by de-registering from the current RH and registering in the new RH [58]. This requires session re-establishment after the handover is completed which can be handled by the Transmission Control Protocol (TCP).

Producer mobility is not as complex as in NDN, since name resolution is done

before the routing process. Moreover, registering to a new RH will force all new requests to be forwarded to the new location. Existing requests will be resent or can be handled by schemes like Mobile IP [96].

### Publish-subscribe Internet Routing Paradigm

PSIRP requires Consumers to subscribe for data to be able to request it. Hence, to support Consumer mobility, Consumers resubscribe for the data needed [99,100]. This forces the Routing function to give a new Forward Identifier for the new location [100].

Similar to NDN, Producer mobility is complex and requires more attention. To be able to reach the publisher at the new location, the topology management function should be aware of the new location.

### Network of Information

NetInf is another architecture that uses early binding. It operates in one of two modes, *name routing* or *name resolution*. This scheme makes Consumer mobility simple like other architectures, where GET packets can be resent to ask for data in the new location.

Provider mobility requires updating the Name Resolution Service (NRS) with the new location change which was claimed to be a scalable process. To support delay-sensitive applications, NetInf can use mobility anchor point mechanisms or late locater construction [30].

Table 2.2: Mobility support in the main four architectures.

| Architecture | Consumer Mobility | Producer Mobility |
|---|---|---|
| **NDN** | Supported by default; Just resend *interest* | Complex, Update the routing state which is not scalable. |
| **DONA** | De-register and Register with the resolution handlers | New Requests: Same as Consumer Mobility. Current request: Resend them or use Mobile IP approach. |
| **PSIRP** | Resubscribe to get new Forward Identifier | Update the topology management function, which is not scalable. |
| **NetInf** | Supported by default; Just resend GET packets | Should update the NRS. For delay-sensitive applications: mobility anchor mechanism can be used. |

### 2.5.3 Discussion

Every new network paradigm must consider node mobility as one of its main features. A comparison on how each architecture attempted to solve mobility is shown in Table 2.2. As shown in the table, handling Consumer mobility is relatively simple compared to Producer mobility which requires more research effort. Mobility handling in some architectures is simplified due to the way naming and routing are handled. For example, the architecture in DONA and NetInf is based on the use of resolution handlers (in DONA) and Network resolution service (in NetInf). These entities are used for early binding of the content's name to the content's location. Hence, locating the node in the network is handled by default in the architecture and no additional mechanism is needed. However, to support fast handover for real time applications, a mobility mechanism such as Mobile IP can be used.

Some architectures (e.g., NDN and PSIRP) need a scalable way to handle mobility. For example in NDN, due to the late binding of content's name to content's location, the Producer mobility is handled by updating each router with the Producer's new

location, which is not a scalable solution. For this reason, many of the proposed mobility mechanisms in the literature are for NDN and some for PSIRP.

In this research, we use NDN as the main architecture to investigate and develop mobility solutions. The reasons behind this are as follows:

1. The impact of mobility is higher in NDN since it is a late-binding network.
2. A simulation tool (ndnSIM) is available as an extension of ns-3 [4].
3. A testbed of 37 nodes in 30+ institutions is running and available to be used by researchers [73].
4. A version of NDN has an early-stage platform (CCNx) which has been already implemented and can be used over the IP network using PCs or Android devices.
5. NDN is one of the most common architectures in the literature to be used as an ICN.

## 2.6 Proposed Mobility Solutions for ICNs

### 2.6.1 Consumer Mobility Schemes

Multiple mobility management schemes have been proposed in the literature to support Consumer mobility and can be classified into two categories: Reactive and Semi-Proactive.

#### Reactive Schemes

The schemes in the first category [65, 83, 105] take actions after the handover occurs when the Consumer connects to the new Point of Attachment (PoA). For example,

in [83] the old PoA will not drop the *data* requested by a mobile Consumer once it receives a handover notification (e.g., disassociation message). Instead, the *data* will be cached until the Consumer connects to the new PoA and retrieves it. The difference between this approach and NDN's is that it asks for the *data* from the old PoA not from the Producer. However, this is only beneficial if the delay to access the Producer is longer than the old PoA.

### Semi-Proactive Schemes

The second category is the semi-proactive based schemes that depend on predicting handover events, and thus mobility supporting actions can be taken in advance. The schemes proposed in [82, 83, 91, 97] predict the future network state to cache the content near the new location and thus the *data* access time is reduced. For example, in [91], the Consumer predicts the movements and notify its PoA with the *interests* that will be missed due to handover. The old PoA in cooperation with the predicted new PoA will cache the *data* in the latter such that it will be ready once the handover is done and the retransmission is issued. The advantage of this semi-proactive approach is that the delay is lower than the reactive and NDN, however, retransmissions are still needed to request the content from the new PoA.

### Discussion

Table 2.3 summarizes the features of the two categories compared to NDN with no mobility scheme in place (Pure-NDN). First, all schemes require *interest* retransmissions to recover the dropped *data*. As discussed, this causes longer delays and extra traffic on the network. Regarding caching, both categories require storing content

Table 2.3: Comparison summary of Consumer mobility schemes.

| **Feature** | **Pure-NDN** | **Reactive** | **Semi-proactive** |
|---|---|---|---|
| *Interest* retransmissions | Yes | Yes | Yes |
| Requires caching | No | Yes | Yes |
| Mobility prediction | No | No | Yes |
| *Data* source | Producer | Old PoA | New PoA |

in network caches to support mobility. Specifically, the reactive schemes cache the content in the new PoA whereas the semi-proactive schemes cache it in the old PoA. Since the new PoA is needed to be identified before handover, predictions are required for the semi-proactive approach.

### 2.6.2 Producer Mobility Schemes

In this section, we investigate several proposed schemes designed to handle Producer mobility in the NDN architecture. Re-applied ideas from current protocols in the Internet such as Mobile IP [19] and DNS were at the core of these schemes. We categorize the mainstream NDN mobility management schemes into: Mobility Anchor, Location Resolution and predictive caching. One management scheme from each approach is selected to be a representative scheme, which will be explained here and evaluated in later chapters.

#### Mobility Anchor Schemes

Schemes that use mobility anchor mechanisms are based on the Mobile IP scheme described in Section 2.5.1. The scheme chooses anchor nodes to be used as Home Agents (HA), where each HA is always aware of the location of its nodes. Moreover, it forwards the *interest*s to the Producer when it is connected to another network (i.e.,

roaming). In this case, the *interests* will not be dropped but may take longer paths to reach the Producer. Moreover, anchors can be installed on Access routers. Examples of schemes that use an anchor node to support mobility are [48, 49, 56, 62, 70, 81].

We choose the scheme proposed in [62] as a representative mobility anchor scheme. The mechanism in [62] solution is based on tunneling. The home content router $CR_h$ is the anchor in this scheme. Moreover, the $CR_h$ forwards any *interest* directed to the mobile Producer to its new location, and when *data* is received it will be forwarded to the Consumer. The scheme uses the following new terms:

1. $PU$: Prefix Update packet sent by the mobile Producer to update the prefix name in the home router.

2. $PACK$: Prefix update Acknowledgment sent by the home router to the mobile Producer in the new location, to acknowledge the prefix update request.

3. *Tentative name prefix*: New prefix to be used in the new location. The tentative name contains the new router prefix plus the old router prefix. The new name should be unique.

4. Interest *(Tunnel)*: The new *interest* generated from $CR_h$ to redirect the original *Interest*. It contains the tentative prefix, the original prefix and the content name.

5. *Data (Tunnel)*: *Data* generated by the mobile source but padded with the tentative prefix and the original name.

The operation of the scheme can be summarized in three main steps:

1. *Movement Indication*:

Once the mobile Producer connects to a new router, it decides on a tentative name prefix that should be unique in the new domain.

2. *Path redirect configuration*:

   The mobile Producer sends PU packet to the router to update $CR_h$ with the new location. The router updates the mobile source record in its routing table, and then reply with PACK. The PU and PACK are handled as regular *Interest* and *Data* packets, respectively. As PU is routed through the network, PIT records are added. PACK follows the breadcrumbs in the PITs to reach the mobile Producer.

3. *Interest Redirection*:

   When an *interest* requesting data from the mobile Producer reaches $CR_h$, it will be encapsulated by a new *interest* (tunneled *interest*) and then sends it to the mobile Producer. The mobile source will send the *data* back to $CR_h$ and the latter will forward it to the requester.

**Location Resolution Schemes**

The approach in Location Resolution schemes is similar to the one used in DNS, where the Consumer queries the location of the Producer before sending *Interest*s. Location Resolution Servers (LRS) are used to resolve location queries, the servers should be always aware of the current location of each node. Examples of schemes that use this approach are presented in [17, 57, 105, 106]. NDN uses late content-to-location binding technique, where the content is matched to a location in the forwarding stage. However, Location Resolution schemes require early binding techniques similar to

DONA [58]. This technique affects content naming used in NDN and requires extra overhead to maintain and query locations.

Kim, et. al have proposed in [57] a mobility management scheme that uses Location Resolution Servers. The operation for both Consumer and Producer is as follows:

1. Producer side: once a handover has occurred, the Producer will update its prefix to match the new location (e.g., Producer named: /prefix_1 moves from access point 1 (AP_1) to access point 2 (AP_2), the Producer's name will change from /AP_1/prefix_1 to /AP_2/prefix_1). This new prefix is then sent to the LRS to update its records.

2. Consumer side: a new timeout period $L_h$ is introduced to detect Producer mobility. When the *Data* takes longer than $L_h$ to reach the Consumer, the latter will query the LRS for the current Producer's location. Once the new prefix reaches the Consumer, it will resend all pending *interests* with the updated location.

### Predictive and Neighbor Caching Schemes

Mobility anchor and location resolution schemes are considered non-predictive schemes, whereas this class includes predictive schemes which solve mobility problems before the handover events. Proposals in [64, 98] are examples of proactive schemes. Research presented in [98] pushes some data to 1-hop neighbors based on probability calculations of the popularity and rarity of the content. However, this approach violates the design of NDN which is Consumer-driven and Push actions are not natively possible in NDN. Additionally, its performance is suboptimal when mobile devices

switch between different technology networks (vertical handover). Moreover, considering only the first-hop neighbors limits the resource utilization by the scheme to successfully manage Producer mobility (i.e., provide local optimal solutions).

The research in [64] follows the same proactive approach but it periodically pushes the data to the vicinity of the Producer such that popular content is always duplicated to nearby routers. However, the scheme fails if the chosen router is not in the path from the Consumer to the Producer (this can happen when a Producer has more than two PoAs). Moreover, both schemes select popular data only to be pushed. However, other less popular data content has to be explicitly handled to provide seamless mobility and avoid the degradation of overall network performance.

#### Other Schemes

Some of schemes are concerned with shortening the routing convergence time such as the proposals in [44, 63, 70, 93]. For example, [44] uses Software Defined Networks (SDN) to update local FIB tables with mobility changes. The scheme in [93] sends routing update advertisement messages to change the FIB entries of routers along the path from Producer to the Consumer. Then, future *interests* hopefully will reach one of these routers and then be redirected to the new location. However, this approach fails when more than two Consumers are requesting from the same mobile Producer, since the update message will stop when it reaches one of the Consumers.

#### Discussion

Schemes in non-predictive category (Mobility Anchor and Location resolution) share the same concept where mapping of location to data is needed to support seamless

mobility. Specifically, the mapping in Mobility Anchor schemes occurs in the Home Router of the Producer; such routers require extra logic to be added. However, Location Resolution need a new entity (LRS) to handle the mapping. Additionally, the latter requires the name of the Producer (i.e., prefix) to be changed after the handover to match its new location. Consequently, the same *data* is stored in the caches with different names which waste the available resources. In the Mobility Anchor Scheme the change in name should not affect the Consumer since the Home Agent handles the tunneling. On the contrary, the Consumers in Location Resolution Schemes should be aware of the change, therefore LRS queries should be sent. As a result, the advantage of late binding strategy proposed in the original NDN will no longer be in use with Location Resolution approach, which could affect other NDN functionalities such as forwarding and routing.

Typically, the tunneling technique used in Mobility Anchor schemes causes longer delays, since *data* packets takes longer paths to reach the Consumer. However, at the same time, this technique minimizes *Interests* being dropped during handover. In Location Resolution schemes some *interests* will be dropped before querying the LRS but the *data* packet takes the shortest path to the Consumer. Generally, mobility anchor approaches have two main problems: a) Single point of failure and b) Triangular routing. On the other hand, Location Resolution has a problem of higher handover latency if the location resolution node is not in the local domain. Finally, the predictive schemes needs prediction algorithms to proactively support mobility. These algorithms are not perfect which may provide false prediction causing wrong caching decisions.

Table 2.4: Comparison summary of Producer mobility schemes.

| Category | Mobility Overhead | | Handover Latency | Other Problems |
|---|---|---|---|---|
| | Consumer | Producer | | |
| **Pure-NDN** | - | Update routers | High | Not scalable |
| **Mobility Anchor** | - | Update proxy agent | Low | Single point of failure and triangular routing |
| **Location Resolution** | Query location | Update server | High | Extra server to handle location queries |
| **Proactive Caching** | Send *interests* to different routers | - | Low | Sensitive to errors in predictions |

All schemes require control packets to be sent in order to support seamless mobility. Such control packets are considered as overhead traffic on the network. In particular, the Mobility Anchor scheme requires a pair of *Interest*/*Data* (prefix update) to be sent for every mobility event to update the Home Agent. On the other hand, Location Resolution Scheme requires two types of overhead. First, prefix update packets which are similar to the mobility Anchor Scheme but will be directed to the LRS. Second, a prefix fetch pair is requested by the Consumer once a timeout period is expired. However, this timeout can occur as result of other events, unrelated to mobility, in the network such as congestion, which causes the scheme to send unnecessary packets. On the other hand, predictive schemes need extra overhead of caching. Since schemes rely on storing *data* on the network to support mobility, cache resources will be used which may affect availability of other content. Table 2.4 summarizes the different categories and the limitations of each one.

## 2.7 Conclusion

In this chapter the ICN paradigm has been introduced and four of the major proposed architectures have been explained. Moreover, the problem of mobility in each architecture was discussed and the state-of-the-art of mobility support protocols were explained and qualitatively compared. We can conclude with the following points:

- The Internet is designed for host-to-host communications. A new paradigm should be designed to change the communication model from host-based to content-based.
- Information-Centric Network is designed for content delivery. There are different ICN architectures proposed in the literature, but all are data centric networks.
- Routing, naming, caching, security and mobility are some of the challenges that any proposed ICN design must take into consideration. Mobility is one of the main challenges in any network, since the number of mobile nodes is increasing.
- The data communication in the current Internet is host-based. In this communication model, mobility support faces problems because the data is bound to hosts' location.
- ICNs offer opportunities to sustain mobility management. These include location independent data, caching and utilizing multiple network interfaces simultaneously.
- Consumer mobility is relatively easier to handle in ICNs, since the paradigm is Consumer-driven. Producer mobility is more complex and needs more research

attention.

- Several solutions have been discussed in this chapter to support both Consumer and Producer mobility.
- We categorize Consumer mobility schemes to reactive and semi-proactive schemes.
- We categorize Producer mobility schemes to Mobility Anchor, Location Resolution and Predictive schemes

# Chapter 3

# NDN Mobility Assessment Framework

We remark that the performance evaluation of NDN mobility management schemes proposed in the literature are mostly performed under simplistic configurations/operations conditions that do not reflect realistic ICN environments. Indeed, there is a lack of comprehensive analysis of the design factors that yield seamless mobility management, and a benchmarking tool to contrast and evaluate mobility management schemes.

In this chapter, we propose a modular mobility assessment framework, developed within ns-3, to evaluate the performance of mobility management schemes under various topologies, heterogeneous Producers and Consumers, and access profiles; which will be released for NDN research. Our goal is to provide the means (assessment framework) and design parameters for researchers in NDN mobility to evaluate the performance of their frameworks, and to benchmark against the already developed and tested schemes.

This chapter is organized as follows. In Section 3.1, the details of the assessment framework is explained. The performance metrics and factors are presented in Sections 3.2 and 3.3 respectively. Section 3.4 explains the experimental setup used in

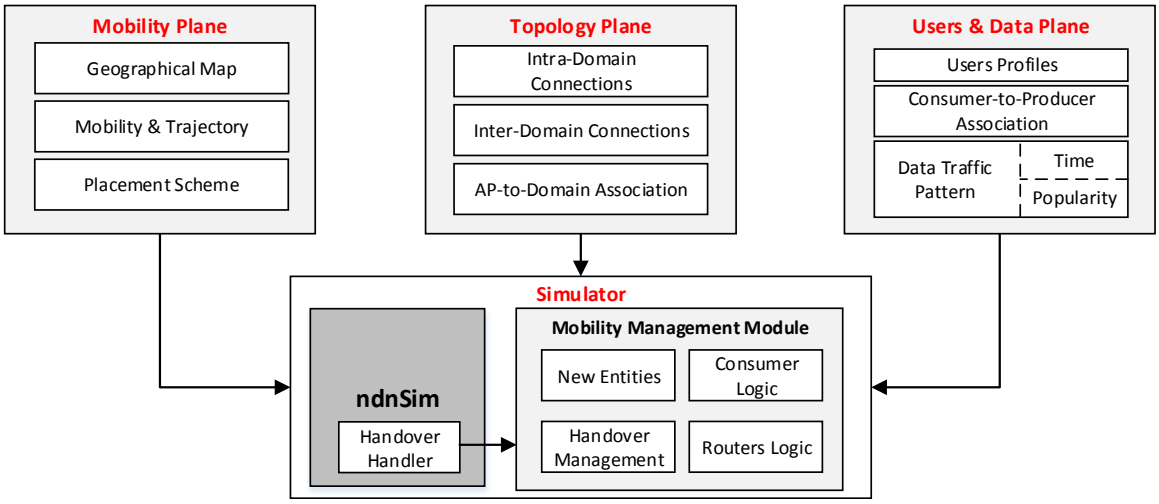



Figure 3.1: The main components of the assessment framework.

this thesis. The design of the technical setup that is used to run the experiments are presented in Section 3.5.

## 3.1 The Design of the Assessment Framework

The proposed framework is designed to be a benchmarking tool for existing and future mobility management schemes. Hence, the tool is used to evaluate and compare the schemes using carefully selected performance metrics. The inputs to the assessment framework are varied to study the performance of the mobility management schemes under different scenarios. The components of the framework are shown in Figure 3.1 and are explained in the following subsections.

### 3.1.1 Simulator

The simulator used in the assessment framework is ndnSIM [4] which is an NDN simulator based on ns-3. In particular, ndnSIM is an event-based simulator implemented in modular C++ classes to model the behavior of NDN entities: FIB, PIT and CS.

Using ns-3 as a base simulator enables many scenarios with different node types and communication technologies such as wired links, Wi-Fi and LTE. Moreover, ndnSIM is an open-source platform which enables researchers to use it as is or extend it with their new contributions.

### 3.1.2 Topology Plane

This component generates various network topologies to be used in the assessment framework. The topologies are designed to be hierarchal with multiple domains such as Transit-stub topologies [87, 88, 101], which allow Intra-domain and Inter-domain communications. Moreover, with hierarchal topologies two different mobility scenarios are investigated: moving to a new Access Point (AP) within the same domain (Intra-domain mobility) and moving to another domain (Inter-domain mobility). Studying both scenarios emphasizes the advantage of data caching during mobility events since the effect of Intra-domain mobility is reduced by the available caches in the domain.

The topology plane generates the network in three steps. First, the domains are generated based on the number of routers and number of distinct domains required. Second, the domains are connected to create the transit network. Third, the APs are associated with the different domains. Figure 3.2 illustrates an example of a hierarchal topology to be used in the assessment framework.

Once the user leaves the communication range of one AP and enters another, a handover event occurs. Specifically, the user dissociates from the current AP and associates with the new AP. With this in mind, APs that are geographically neighbors, are not necessarily neighbors in the network topology (i.e., within the same domain),

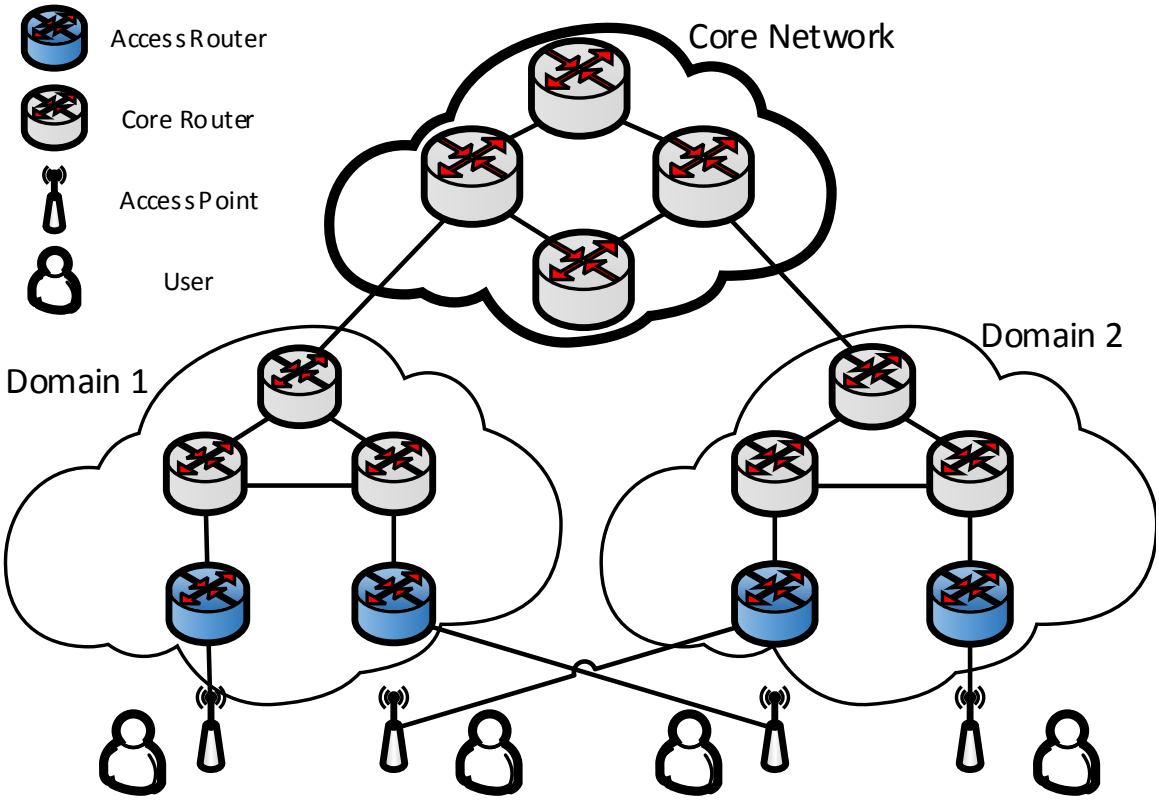



Figure 3.2: A transit-stub hierarchal topology highlighting multiple domains and wireless access points

as shown in Figure 3.2.

### 3.1.3 Mobility Plane

The positions and movements of users on the map are the outputs of the mobility plane. The map may consist of streets, sidewalks and/or buildings. Urban areas, which are characterized by having higher densities of users and Wi-Fi APs, are examples of geographical maps that can be used. This component also places the APs on the map in addition to the fixed users such as PCs and servers.

Users' trajectories can be generated using analytical random mobility models,

simulation models or real mobility traces. Random Waypoint, Random Walk and Gauss-Markov are examples of analytical models that use mathematical calculations to determine the next position of a node. On the other hand, simulation models such as traffic simulators provide more realistic user movements within a given map. Simulation of Urban MObility (SUMO) [59] is widely used to generate such trajectories for vehicles and pedestrians moving at various speeds. Finally, Mobility traces of real user's movements can be captured and used in the assessment framework similar to the dataset published in [31].

### 3.1.4 Users and Data Plane

This component determines the traffic profile of each user, what data the Producers can generate and what (and when) the user requests data. Specifically, Consumers' *Interest* patterns are generated which include the time of request, the Producer's name and the *data* name. To create realistic request patterns, different *data* are requested with different popularity (i.e., some *data* are requested more frequently than others).

To model content popularity, distributions such as Zipf's law [42] can be used. The distribution is controlled by parameter $s$, where lower values of $s$ give more uniform-like distributions (i.e., if $s = 0$ then all *data* has the same probability). The probability mass function of this distribution is given by Equation 3.1.

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)} \tag{3.1}$$

Where $k$ is the rank of the *data*, $N$ is the size of the population. Figure 3.3 depicts a few samples of the Zipf distribution that are generated by this component.

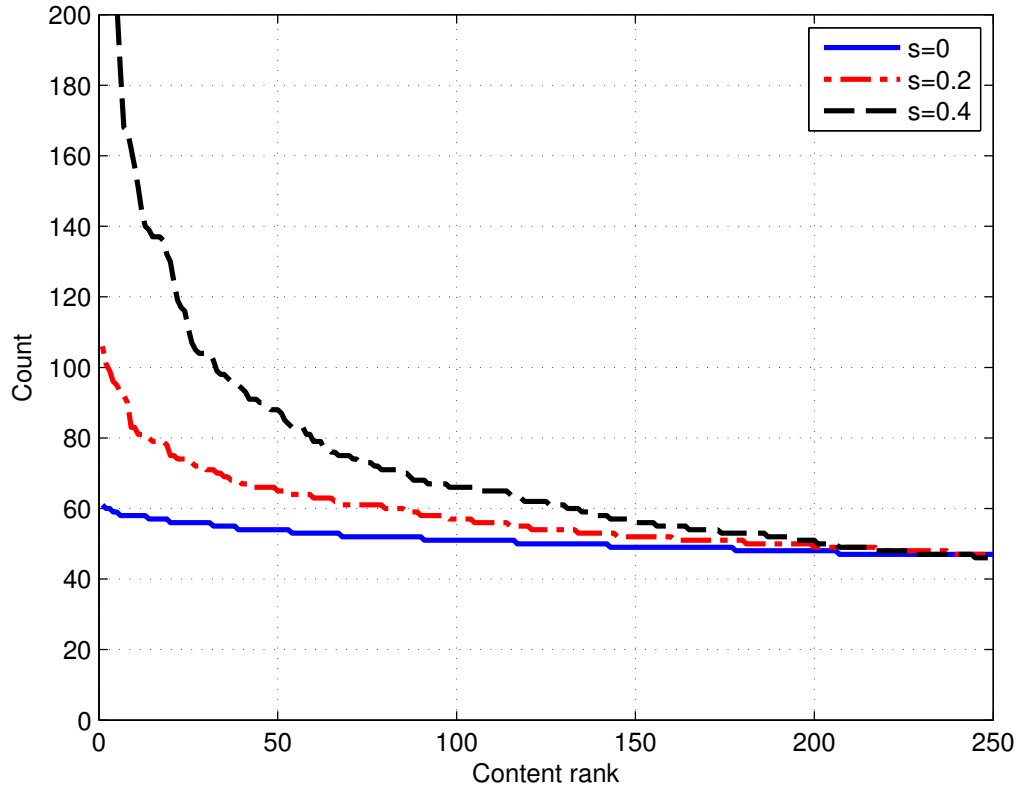



Figure 3.3: The Zipf distribution with different values for parameter $s$.

### 3.1.5 Mobility Management Schemes Implementation in ndnSIM

The existing ndnSIM (and ns-3) does not provide Wi-Fi handover between access points, nor an NDN mobility scheme. Therefore, two main updates are done to the current ndnSIM: AP handover handler and a template for mobility management schemes. A new class called `HandoverHanlder` is created and added to the Mobility Module of ns-3. For each mobile node, an object of this class is instantiated and aggregated to class `Node` of ns-3. The handler is responsible for detecting the handover event based on the power received from the AP or the distance between the AP and the user. Specifically, it will dissociate the mobile node from the old serving AP and

associate it to the new AP. The class triggers two events in the mobility management scheme by calling the function `PreHandover` before dissociating from the old AP and `PostHandover` after associating with a new AP.

`MobilityManagement` is a base class for any mobility management scheme to be implemented in ndnSIM. This abstract class contains two pure virtual functions (i.e., the derived class is required to implement this function) `PreHandover` and `PostHandover`. In particular, these functions implement the required actions to be done by the mobile node before and after the handover. Moreover, the class may include any extra logic to be added to the Consumers, routers and new entities. For example, Location Resolution Schemes need a new timeout period to be added to the Consumer application and the logic of the LRS. `MobilityManagmentHelper` is a class that is used to aggregate the `MobilityManagement` on all nodes for handling Producer mobility.

## 3.2 Performance Metrics

In order to provide a seamless experience during mobility in NDN, the management scheme needs to keep the delay experienced by the Consumer low, avoids *interest* retransmissions and minimizes the overhead. Measuring the following metrics provides a complete picture of the scheme's performance which covers the effect on both the user and the network.

1. *Consumer Delay* is the total delay that Consumer experiences to request an *interest* and receive the corresponding *data*. Specifically, it is the time difference between the first attempt of sending the *interest* and receiving the *data*. This includes the total timeout and the delay of retransmitted *interest* and *data*

packets

2. *Delivery Ratio* is the proportion of successful *data* packets received by the Consumer to the total number of *interest*s sent. This metric is a measure of how successful the scheme is in avoiding both *interest*s and *data* drops.

3. *Scheme Overhead* is calculated as the percentage of the total number of control packets generated by the scheme to the total number of Interests. For example, in Location Resolution schemes, the overhead includes the prefix update and query packets discussed in Section 2.6.2.

4. *Retransmissions ratio* is the ratio of the total number of retransmitted *interests* to the total *interests* sent by Consumers.

5. *Cache Hit Ratio* of a router is the proportion of successful *data* retrieval from the cache of the router (hit) to the total *interests* passing through it.

There are other metrics that can be used to evaluate schemes, such as throughput and cache content diversity. However, we choose the above mentioned metrics which are sufficient to highlight performance changes in NDN.

## 3.3 Experimental Factors

There are many parameters that can be changed in the assessment framework. However, we choose the factors that impact the performance metrics the most which we then study and evaluate. The factors studied are:

1. Mobility percentage:

This factor represents the ratio of mobile users to the total number of users in the network. There are two types: Consumer mobility and Producer mobility percentages. For instance, a network of 100 users (50 Consumers and 50 Producers) with 50% Consumer mobility and 20% Producer mobility means that there are 25 mobile Consumers, 10 mobile Producers and the remaining are stationary. Clearly, increasing percentage of mobility has more impact on the network, hence the effectiveness of the mobility scheme can be shown.

2. Cache Size:

   Caching is a core feature in NDN, which reduces the latency and traffic load in the network. Obviously, bigger cache sizes allows more *data* to be stored in the network and hence less impact of mobility. For instance, if the cache size is infinite (compared to the content), any requested *data* will be cached and nothing will be replaced. Hence, all future repeated *interests* will be satisfied from the routers, not from the Producers.

   The effectiveness of the cache size is very related to the number of distinct content that can be produced in the network. Therefore, we represent the cache size as a percentage of the total possible content. For example, 1% cache size with 10,000 distinct content, means that the cache size is 100 units assuming that all content has a fixed size.

3. Content popularity:

   As mentioned before, popularity is modeled using a Zipf distribution which is controlled with a parameter $s$. Increasing the value of $s$ provides a distribution

where less content is more popular. This fact combined with caching can increase the performance of the network and hence reduce the impact of mobility.

4. Prediction Errors:

   Some schemes require predicting future information to operate. Accordingly, faulty predictions can provide lower performance. Therefore, the proposed schemes are tested under percentage of error in predictions to evaluate its robustness. More details about this factor are discussed in Chapters 4, 5 and 6.

## 3.4 Experimental Setup

In this section, we present the setup of experiments conducted to get the results shown in this thesis. Table 3.1 summarizes the fixed simulation parameters used.

### 3.4.1 Topology

The topology used is transit-stub network that consists of 40 core routers and distributed into 5 domains. Moreover, core routers are connected with links of 10Mbps capacity, while access routers are connected to the core routers with 5Mbps links which have a propagation delay of 10ms. These settings represent the share of resources assigned to the users in the simulation.

The users move in a map that is assumed to be 7x7 street grid plan (e.g., Manhattan) where users are either pedestrians on sidewalks or riding vehicles (private cars or public transit). Moreover, Wi-Fi APs are installed at every intersection to provide Internet access within its coverage range which its radius is assumed to be 200m . Additionally, Wi-Fi 802.11g is used as a standard for the wireless medium.

Table 3.1: Simulation parameters used in performance evaluation experiments.

| | Parameter | Value |
|---|---|---|
| **General** | Simulation duration | 1000s |
| | Transit period | 80s |
| | Map size | $1400m \times 1400m$ |
| | Number of blocks | $7 \times 7$ |
| | Number of users | 100 |
| | Producers | 50 |
| | Consumers | 50 |
| **Application** | *Interest* rate | 50-80 $I$/s |
| | Zipf's parameter $s$ | 0.2 |
| | Content per Producer | $1000 \times 1KB$ |
| **Topology** | APs | 49 |
| | AP range | 200m |
| | Number of routers | 40 |
| | Core router's links | 10Mbps |
| | Access router's links | 5Mbps |
| | Propagation delay | 10ms |
| **Mobility** | Model | Manhattan |
| | Handover delay | 0.5s |
| | Speed | 70 km/h |
| **NDN** | Forwarding scheme | BestRoute |
| | Cache replacement | LRU |
| | Cache size | 1000 objects |

Furthermore, the handover delay (dissociating and associating to APs) is assumed to be 500ms.

### 3.4.2 Users and Request Patters

There are 100 users, split into 50 Consumers and 50 Producers. Each Producer provides 1000 unique *data* objects (1KB each), and content popularity is modeled using Zipf's law distribution with $s = 0.2$ (unless it is changed in popularity impact

testing). Consumers request *data* by sending *interests* with a random rate (frequency) between 50 and 80 *interest* per second. Moreover, the interval between *interests* is random that follows exponential distribution with $mean = 1/frequency$.

### 3.4.3 NDN

The main functionalities of NDN are assumed to have the default values in ndnSIM. In particular, ndnSIM uses an opportunistic scheme named as "Cache Everything Everywhere" for object caching [29]. NDN nodes using this scheme cache any *data* passing through it. In the case of full cache, Least Recently Used (LRU) replacement policy is used. The default cache size in the simulation environment is designed to be 2% of the total content in the network; therefore the cache size = 50 (Producers) × 1000 (content per Producer) × 2% = 1000 *data* objects. The forwarding strategy *Best Route* is used in ndnSIM where the next hop is decided on the best-calculated metrics such as the number of hops, delay and congestion.

## 3.5 Technical Setup

This section explains how the experiments are run and the results are collected. The setup has the following components as shown in Figure 3.4:

1. Workers nodes:

   The simulations are run on these nodes which are Linux machines prepared with the following softwares:

   (a) ndnSIM [4]: the network simulator.

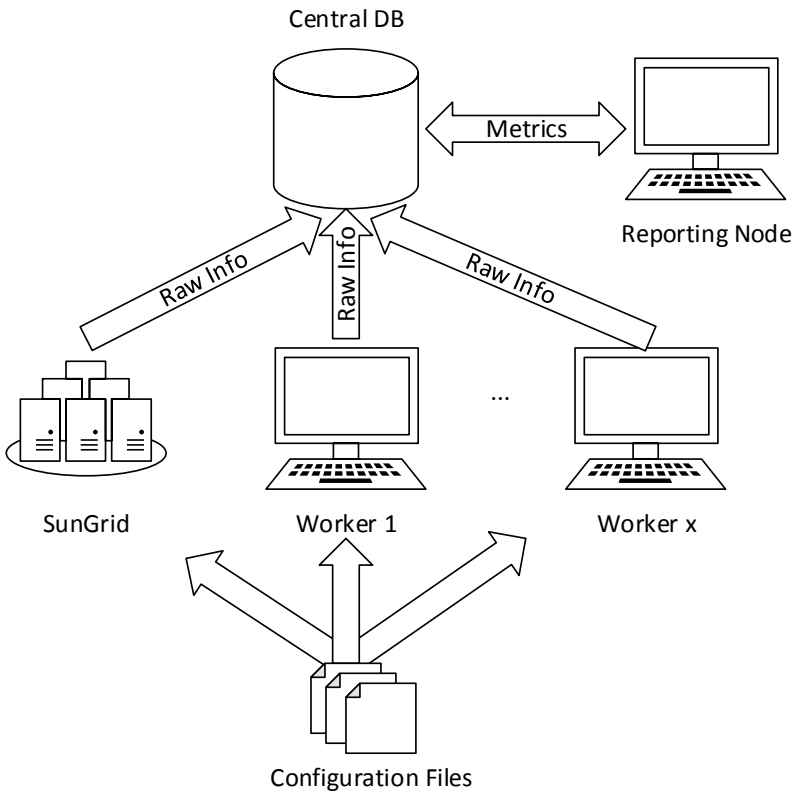



Figure 3.4: The technical setup used to run experiments using the assessment framework.

(b) SUMO [59]: the traffic simulator that generates the mobility traces to be used in ndnSIM.

(c) SQLite [92]: a local database to store progress information and statistics.

(d) Gurobi [47]: an optimization solver used to provide solutions for some proposed schemes.

2. Central database:

   After each run of an experiment, the computing node stores the calculated metrics in a central MySQL database.

3. Reporting node:

   A node used to generate reports and plots of the experiments using MATLAB.

One of the main computing nodes that were used in this work was the high performance computing grid which is provided by the Centre of Advanced Computing (CAC) [20]. The grid enabled us to execute 80 runs at a time which decreased the simulation time.

### 3.5.1 Random Runs

For each experiment, the simulations are executed for enough random runs such that the results are statistically significant. In particular, random variables were used extensively in ndnSIM and changing the random seeds in every run will generate a new instance of the simulation. Moreover, the average and 90% confidence interval were calculated out of all runs. We found that all results reported have a maximum deviation of 5% from the reported average values.

### 3.5.2 Experiment Running Work Flow

The experiment starts by submitting a configuration file to the worker node. The file contains the simulation parameters, the factors to be changed and the number of random runs. Moreover, the worker node creates a new experiment in the central database and then creates a simulation for each combination of factors stated in the input file. Each simulation is then run multiple of times with a different random seed for every run.

## 3.6 Summary

There are existing mobility schemes in the literature that claim to support seamless mobility, but have never been investigated or compared to a benchmark. Therefore, we designed and implemented a novel assessment framework to be used as a benchmark tool and made available to NDN researchers to investigate existing and future mobility management schemes. The framework is used to evaluate proposed schemes and compare it to the one proposed in the literature.

# Chapter 4

# Consumer Mobility Benchmark Scheme

The implicit assumption, in NDN's design, that mobility is supported intrinsically by *interests* retransmissions is found to be suboptimal. This increases the data access time and degrades the network's total throughput. Therefore, the design of a mobility management scheme for both Consumers and Producers is inevitable to a successful NDN. As mentioned in Section 2.6.1, the existing proposed schemes for Consumer mobility use reactive or semi-proactive techniques to recover after a mobile event. In this chapter, we consider a fully proactive approach to support seamless Consumer mobility before handover.

We propose a design for a proactive mobility scheme, *OpCCMob*, that exploits Consumers' location predictions and data request patterns to find the optimal data placement in network caches and avoid data drops and retransmissions. The contributions in this chapter are summarized as follows:

1. We formulate the problem of the content placement based on the forecast Consumers' locations. The formulation minimizes the total Consumer's delay while considering network overheads associated with data placement and removal.

2. An optimal benchmark solution is then obtained using Gurobi optimization solver [47], which is essential to evaluate existing and future NDN mobility management schemes.

3. In the light of the NDN compliant assessment framework from Chapter 3, we define the performance gains due to using location predictions in mobility supporting schemes.

The remainder of this chapter is organized as follows. An overview of the proposed benchmark is explained in Section 4.1. The system model and formulation of the optimal caching scheme are presented in Sections 4.2 and 4.3, respectively. Our simulation experiments and results are discussed in Section 4.4. We conclude our findings in Section 4.5.

## 4.1 Scheme *OpCCMob* Overview

A moving Consumer will not be able to receive the data requested just before the handover event, since it will be disconnected from the old PoA. Hence, the Consumer has to retransmit the *interests* after connecting to the new PoA. However, if the data is cached near the Consumer such that it can be retrieved before the handover, the data will not be dropped. This is the core idea of *OpCCMob*. In particular, *OpCCMob* proactively chooses the optimal routers to cache the required data such that no retransmissions are needed and while maintaining the overhead on the network. Table 4.1 highlights the main differences between the proposed scheme and other approaches in NDN Consumer mobility.

The proactive approach requires a knowledge of the future to operate seamlessly. In particular, the scheme needs to detect the Consumer mobility event before it occurs,

Table 4.1: *OpCCMob* compared to the state-of-the-art Consumer mobility schemes.

| **Feature** | **Pure-NDN** | **Reactive** | **Semi-proactive** | ***OpCCMob*** |
|---|---|---|---|---|
| *Interest* retransmissions | Yes | Yes | Yes | No |
| Requires caching | No | Yes | Yes | Yes |
| Mobility prediction | No | No | Yes | Yes |
| Data prediction | No | No | No | Yes |
| Data source | *Producer* | Old PoA | New PoA | Old PoA |

in addition to the future requests that will be sent during handover. We remark that mobility and content are highly predictable due to patterns of user movements and content requests as will be discussed in Section 4.1.1. In this chapter, we assume that this information is available and correct (i.e., perfect knowledge).

*Interests* requested just before handover will not be satisfied, since by the time the data reaches the previous PoA, the Consumer will be connected to the new PoA. We assume that the handover of a mobile Consumer occurs at $t_{HO}$ and that $RTT$[1] is the average data retrieval time. The *interests* requested by this user between $t_{HO} - RTT$ and $t_{HO}$ are predicted to be dropped and will be retransmitted, see Figure 4.1. The goal of *OpCCMob* is to reduce this period (i.e., reduce $RTT$) by caching the data near the Consumer.

The time horizon is divided into small time slots where the data placement at each time slot is considered as a separate optimization problem. Given the set of predicted requests of all mobile Consumers in one time slot, we can define the problem as finding the optimal placement of data in network caches such that no retransmissions of *interests* are required by mobile Consumers and with bounded overhead on the network.

[1]Round Trip Time

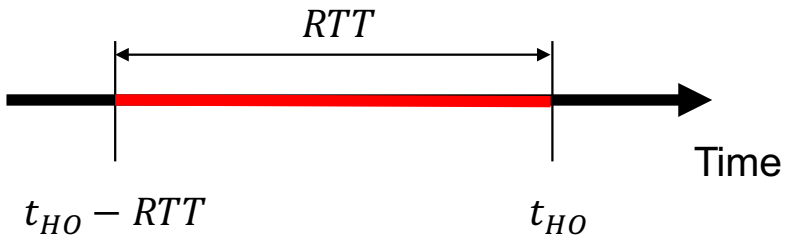



Figure 4.1: Consumer mobility timeline. *Interests* requested at $[t_{HO} - RTT, t_{HO}]$ need retransmissions.

Mainly, the overhead can be of two types:

1. Path updates: Data can be placed on any cache in the network. Caching on a router that is not on the path between the Consumer and a Producer requires a route to be created temporarily from the Consumer to the router chosen. Hence, an overhead to update the FIBs of the new path is needed.

2. Cache cost: Adding data to a full cache requires replacing some cached items based on the replacement policy in place. Both the total number and value of removed data should be bounded to avoid affecting other users.

### 4.1.1 Prediction tools

To proactively identify the *interests* that may not be satisfied due to handover, two pieces of information are needed in advance. We need to predict the time of handover of mobile users and the content that will be requested in the near future.

#### Mobility Prediction

Handover events can be foreseen using the network topology and the location of the user which can be obtained by using location predictors as in [2, 3, 9, 60, 90]. These

tools adopt location history, current position and trajectories to predict the future user locations in the network. Using the estimated location, the knowledge of network topology and user access rights, the user current and future PoAs and the nearest caching routers can be determined. Since the scheme acts proactively, before the user actually moves, the future PoA is not required to be predicted. In particular, the scheme needs to know whether the user will change its PoA in the near future or not.

#### Data Prediction

The second piece of information is the Consumer requests pattern, which can be predicted using various techniques such as reservoir computing, stochastic models of user behavior and biology-inspired survival analysis [8, 23, 28, 77, 104]. Users preferences, historical request patterns and early popularity measures are used in such tools to predict future requests. These tools are currently used in the Internet in different applications such as YouTube video recommendations and Facebook news feed.

### 4.1.2 Special Cache Design

Based on the replacement policy [2] content can be replaced by new data. To avoid this replacement, the proactive data must be treated differently so that it is not replaced by other content before the Consumer makes the request. Therefore, a split cache design is proposed to serve this purpose.

Namely, the split cache is a regular cache with a reserved space for mobile requests. Figure 4.2 shows the design of the special cache where the data of the basic and the proactive data are separated. Furthermore, the separation can be resized to

[2] The policy that decides which content to be removed in order to store a new content (e.g., Least Recently Used (LRU), Least Frequently Used (LFU) and First In First Out (FIFO))

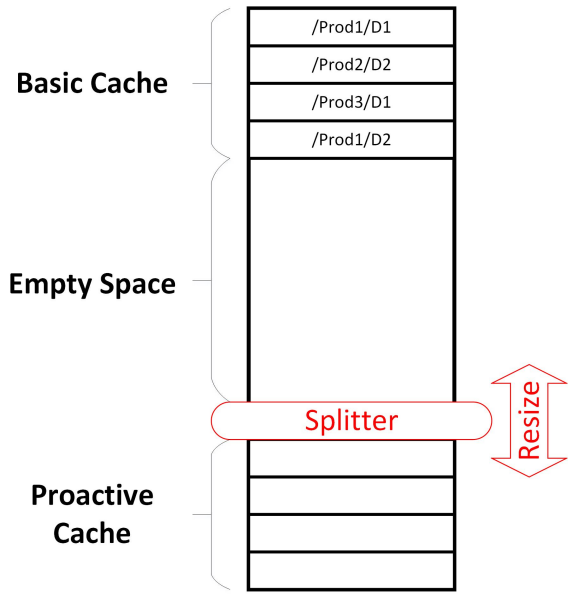



Figure 4.2: The proposed design of the special cache to store proactive data without replacement.

accommodate proactive data without violating the cache capacity. In particular, once new proactive data arrives, the cache stores it in the proactive part. However, if it is full, it will be resized by replacing one item from the basic cache. Figure 4.3 show the decision logic to cache data in the new design.

## 4.2 System Model

In this section, we define the system model used to formulate the problem. Let $U$ be the set of users in the network. Each user $u \in U$ can be either a Consumer or a Producer for data $d \in D$, where $D$ is the list of all data items in the network. Moreover, $u(d)$ is the Consumer of data $d$, whereas $u'(d)$ is the Producer of the same data.

The network topology is represented by a graph $G = (V, E)$, where each node in

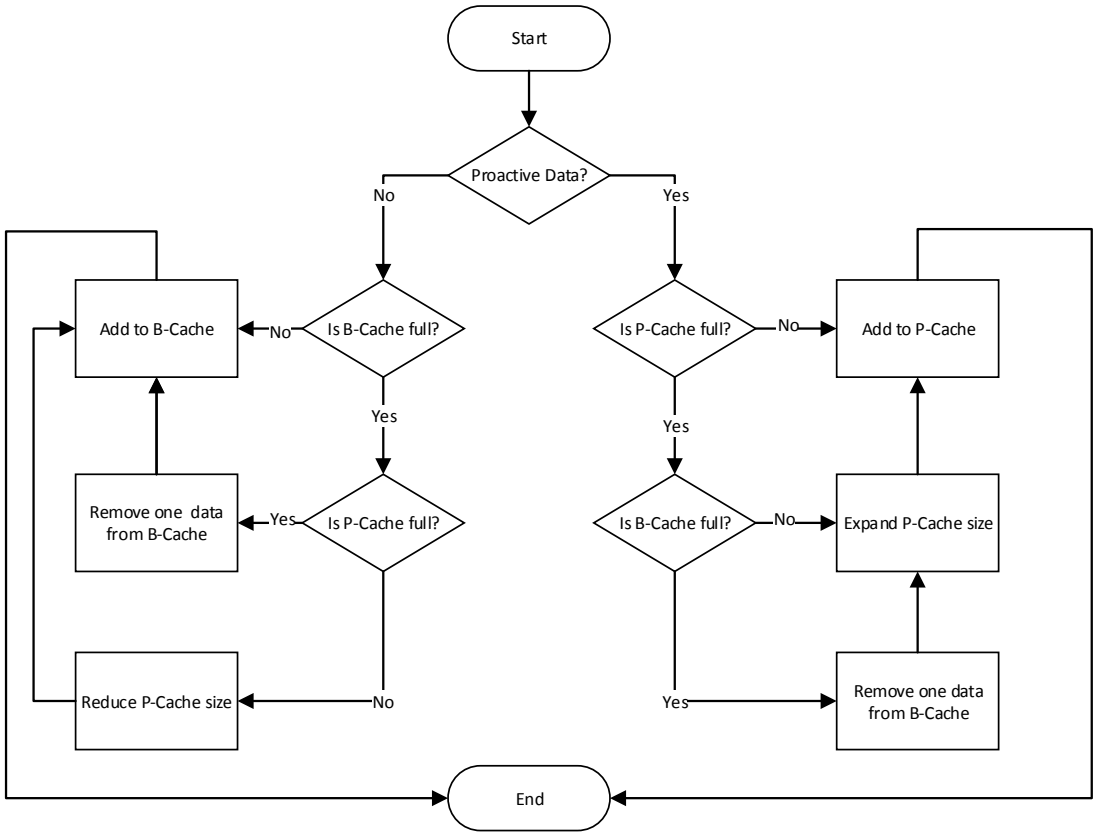



Figure 4.3: The logic used to cache new data in the special cache.

$V$ is either a user or a router. Each router $r$ in the set of routers $R$ has a current cache size denoted by $c_r$ and a maximum cache capacity of $c_r^{max}$. The special space reserved for proactive caching has a maximum capacity $C_r^{SpMax}$ and a current capacity $C_r^{Sp}$.

Let the path taken by an *interest* or *data* from node $x$ to node $y$ is denoted by a sequence of nodes $P_{x \to y}$. Accordingly, the cardinality $|P_{x \to y}|$ represents the number of hops traversed using this path.

Each data item $d$ has a content value $\epsilon_d$ which can represent the content popularity, Producer's quality, application type or user's priority. In this work, $\epsilon_d$ is assumed to be the popularity of an item $d$ (i.e., $\epsilon_d$ =probability that $d$ will be requested next).

## 4.3 Problem Formulation

Finding the best possible placement of data with bounded overhead can be formulated as an optimization problem with two decision variables that determine where to place the data and which data to remove. The first decision variable $\delta_r^d$ is 1 if the solution decides to cache data $d$ in router $r$. The second decision variable is $\rho_r^m$, which decides on how many items should be removed from the cache to provide a space for the new data. In particular, if $\rho_r^m$ is 1, then $m$ items will be removed from router $r$. The formulation is as follows:

$$\min_{\delta_r^d, \rho_r^m} \sum_{d \in D} \sum_{r \in R} |P_{u(d) \to r}| \delta_r^d \tag{4.1}$$

S.t.

$$\sum_{r \in R} \delta_r^d = 1 \quad \forall d \in D \tag{4.2}$$

$$\sum_{r \in R} \mathbb{P}_d^r \delta_r^d < \sigma \quad \forall d \in D \tag{4.3}$$

$$\sum_{m=0}^{c_r} \rho_r^m = 1 \quad \forall r \in R \tag{4.4}$$

$$c_r + \sum_{d \in D} \delta_r^d - \sum_{m=0}^{c_r} m \rho_r^m \leq c_r^{max} \quad \forall r \in R \tag{4.5}$$

$$\sum_{d \in D} \delta_r^d - \sum_{m=0}^{c_r} m \rho_r^m \geq 0 \quad \forall r \in R \tag{4.6}$$

$$\sum_{m=0}^{c_r} m \rho_r^m < \alpha \quad \forall r \in R \tag{4.7}$$

$$(\sum_{m=0}^{c_r} \rho_r^m k_r^m - \sum_{d \in D} \epsilon_d . \delta_r^d) / K_r < \omega \quad \forall r \in R \tag{4.8}$$

$$c_r^{Sp} + \sum_{d \in D} \delta_r^d \leq c_r^{SpMax} \quad \forall r \in R \tag{4.9}$$

The objective function 4.1 is to minimize the delay of data retrieval from the chosen router. The delay calculated as the total number of hops to reach the router ($|P_{u(d) \to r}|$). Constraint (4.2) ensures that each data will be cached only once in any of the routers. The overhead of updating the path is bounded by threshold $\sigma$ and Constraint (4.3) whose left-hand side represents the number of path updates for each data $d$. In particular, $\mathbb{P}_d^r = |P_{u(d) \to r}| - |P_{u(d) \to u'(d)} \cap P_{u(d) \to r}|$ is the number of non-common edges between $P_{u(d) \to u'(d)}$ (path from Consumer $u(d)$ to Producer $u'(d)$) and

$P_{u(d)\rightarrow r}$ (new path from Consumer $u(d)$ to chosen router $r$).

Constraint (4.4) is used to force caching each data exactly once. Constraints (4.5) to (4.9) are related to caching and data replacement. In particular, (4.5) is used to ensure that the number of items in the cache does not exceed the maximum cache capacity. Constraint (4.6) minimizes the replacement by only allowing data removal from the cache if it is full. $\alpha$ is a threshold to bound the total number of replacements per router using (4.7), and thus it limits the cache cost overhead. Since some content will be replaced, the total content value of the router will change. The reduction of this value is bounded by a threshold $\omega$ in Constraint (4.8).

The value of variable $k_r^m$ is the total content value of $m$ items to be removed from router $r$, and $K_r$ is the total content value of all the items in $r$. Since the data in the special cache should not be replaced, the amount of data to be stored in a router should not exceed the number of empty slots in the special cache. Such a restriction is handled by Constraint (4.9).

The objective function and all the constraints are linear and all the decision variables are binary. Therefore, the problem is a 0-1 Integer Linear Program (0-1 ILP) whose optimal solution can be obtained using branch and bound methods implemented in commercial solvers.

## 4.4 Results and Discussion

### 4.4.1 Implementation of *OpCCMob* in the Assessment Framework

The optimal Consumer mobility management scheme is implemented by using the aforementioned *MobilityManagement* module as a base class. The new module's functionalities are:

1. Predict the *interests* that will be dropped due to mobility events. Since perfect knowledge is assumed for finding the optimal solution, the new module will use the information provided from the Mobility and User planes to find these *interests*.

2. Create a Gurobi[3] model of the formulation proposed in Section 4.1. The current state of the network is used to find the thresholds and constraints' boundaries.

3. The model is solved periodically for every time slot (optimization Window) considering the suspected *interests* to be dropped in the next slot.

4. The solution that Gurobi finds is then applied to the network. For every $\sigma_r^d = 1$, data $d$ will be cached in router $r$, using the special cache space mentioned earlier. If $r$ is not on the path from the Consumer to the Producer (i.e., $r \notin P_{u(d)\rightarrow u'(d)}$), FIB entries are added to every non-common router in $P_{u(d)\rightarrow r}$.

Following these steps ensures that all *interests* issued just before a handover event will be directed to a stored data in the special caching space chosen by the optimizer. Hence, no retransmission is required.

### 4.4.2 Experimental Setup

The experimental setup used is explained in Section 3.4. Additionally, *OpCCMob*'s parameters $\alpha$ and $\omega$ are both set to 10% which control the value and number of replacement in NDN caches. This limits the impact on other data, since increasing the

[3]Gurobi is a state-of-the-art mathematical programming solver. It exploits multi-core processors and the latest algorithms to solve optimization problems. Gurobi is configured by using different modeling languages such as AMPL, GAMS and MPL. Moreover, optimization models can be created using programming languages such as C++, Java and Python. We use Gurobi 6.5 integrated with ndnSIM where the proposed formulation is modeled using C++.

Table 4.2: Simulation parameters of *OpCCMob*.

| Parameter | Value |
|---|---|
| $\alpha$ | 10% |
| $\omega$ | 10% |
| $\sigma$ | 20% |
| Optimization window | 2 seconds |

percentage will allow more regular data to be replaced. Furthermore, the optimizer runs every 2 seconds to find the optimal placement and apply it to the network. Short optimization window creates many smaller optimization problems. Whereas large windows, create fewer complex problems. The parameters are summarized in Table 4.2.

### 4.4.3 Comparative Schemes

We study *OpCCMob* and compare it to three other schemes under different scenarios. The first scheme is NDN with no Consumer mobility scheme while *interests* retransmissions are solely used (it is referenced as Pure-NDN). The second scheme is proposed in [83] and it is used to represent the Reactive schemes. Lastly, the third scheme is Semi-proactive which is simulated according to the work in [91]. Proactive schemes usually predict more than one future location to adapt to uncertainty. However, in the following experiments (except the last one), we assume the best case scenario where the future PoA is correctly predicted.

### 4.4.4 Consumer Mobility

In the first experiment, we evaluate the effectiveness of the aforementioned mobility schemes under various percentages of Consumer mobile nodes in the network. Figure

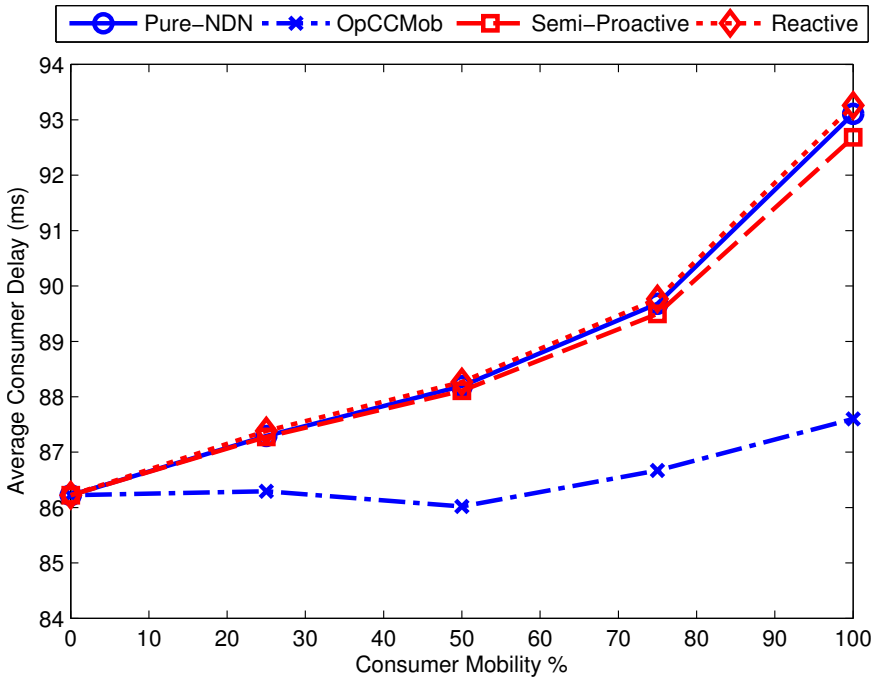



Figure 4.4: *OpCCMob* compared to other schemes under different mobility percentages, showing average Consumer delay over all *interests*.

4.4 shows the ability of the optimal scheme *OpCCMob* to relatively mitigate the effect of mobility on the average Consumer delay and provide stable performance. In particular, proactively caching the required data near the Consumer has avoided retransmission, hence the delay is not increased (stays at 86-87ms).

Comparatively, Pure-NDN suffers from longer delays which have increased by 8%. This is the result of the simple approach of NDN which relies on *interests* retransmissions. Moreover, the performance of the two schemes representing the main approaches proposed in the literature are close to pure-NDN. For further comparison, Figure 4.5 and Table 4.3 show the metrics for the *interests* sent during handover (i.e., *interests* sent in $[t_{HO} - RTT, t_{HO}]$). The delay of the Reactive technique is relatively longer than Pure-NDN by 2ms. This is due to the approach taken by the Reactive

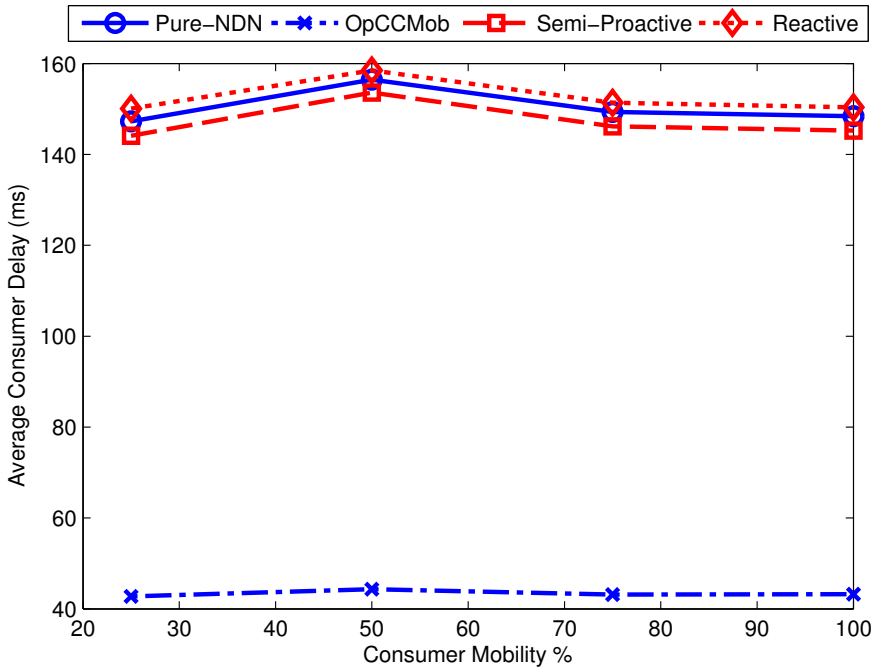



Figure 4.5: *OpCCMob* compared to other schemes under different mobility percentages, showing average Consumer delay for *interests* issued in $t_{HO} - RTT \longrightarrow t_{HO}$

scheme where the retransmitted *interests* are directed to the old PoA (instead of the Producer as in Pure-NDN), assuming it can be closer to the new PoA which is not always true, especially in cases where Consumers move to a new domain. On the other hand, the Semi-proactive scheme has successfully reduced the delay by 2% compared to Pure-NDN. The limited improvement is the result of relying on *interests* retransmissions, and under utilizing caching resources while storing the data. This can be shown as well in the delivery ratio (Table 4.3) where it is very similar to Pure-NDN. On the contrary, the benchmark *OpCCMob* has a higher delivery ratio (95%) since no retransmissions are needed.

While *OpCCMob* provides lower and upper bounds for the Consumer's delay and delivery ratio, respectively, the gap between the former and each of the other schemes provides room for enhancing their performance. For instance, the semi-proactive

Table 4.3: Average delivery ratio and scheme overhead of all Consumer mobility schemes.

| Scheme | Delivery Ratio for mobile *interests* | Overhead message per *interests* |
|---|---|---|
| **Pure-NDN** | 89% | 0 |
| ***OpCCMob*** | 95% | 1.06 |
| **Semi-Proactive** | 89% | 1 |
| **Reactive** | 89% | 1 |

scheme can consider other caching resources besides the PoA of the Consumer. However, the optimality of *OpCCMob* requires extra overhead to deliver the data near the Consumer and to update the path to the chosen router. Table 4.3 compares the overhead of all schemes. *OpCCMob* has an extra 6% overhead over other schemes. This is the result of the path update messages needed to support off-path caching.

### 4.4.5 Cache Size and Content Popularity

Mostly all the considered mobility management schemes utilize the cache resources available in the network. In this experiment, the effect of the cache size is studied by varying the percentage of cache sizes to the total available content (from 0.4% to 6%). Figure 4.6 demonstrates how the delay is reduced, in each scheme, when more data can be cached in the network. In the case of small cache sizes in 0.4% (200 cached items out of 50,000 total items), the gap to the optimal scheme increased by 50% since it caches in many routers and not is not limited to one router. Consequently, the overhead to update off-path routers is increased by 72% (i.e., on average 1.8 packets are needed to cache one data). Fixing the cache size and changing the content popularity provides the same behavior as varying the cache size as shown in Figure 4.7. As the factor $s$ increases, fewer data get more popular and hence the

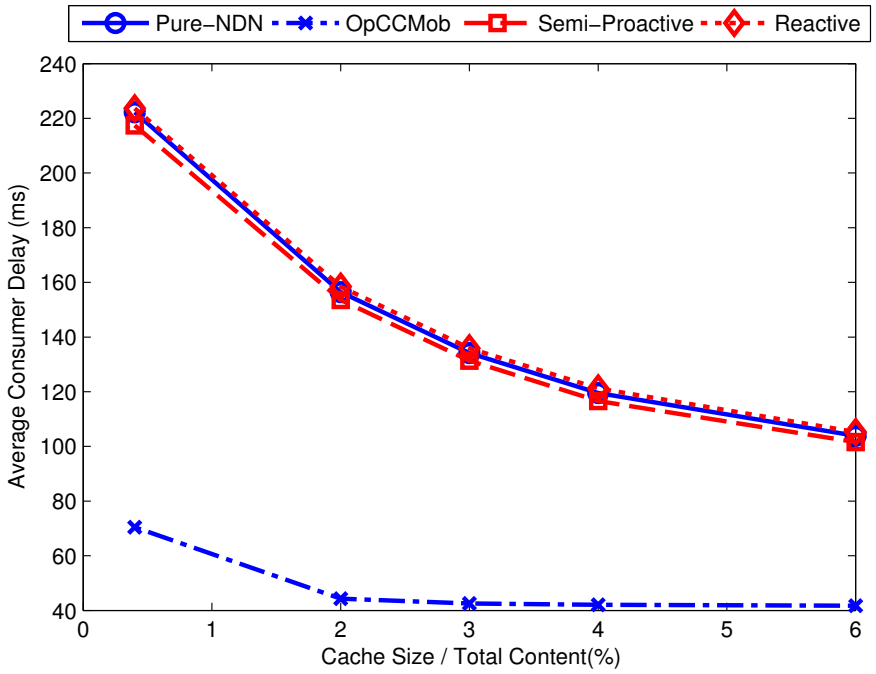



Figure 4.6: *OpCCMob* compared to other schemes under different cache percentages, showing average Consumer delay at 50% mobility.

caches are able to store the data requested the most. For instance, when $s = 1$, the number of frequent requests can fit in network caches, therefore the schemes including Pure-NDN have a similar performance.

### 4.4.6 Sensitivity Analysis

In the previous experiments, we assume that the Semi-proactive schemes and *OpCC-Mob* adopt error-free predictions. To measure the impact of imperfect prediction in mobility and requests, we conduct the following two experiments.

#### Imperfect Mobility Prediction

Error in predicting the future location of a node may lead to incorrect PoA. To simulate such errors, we add a uniformly distributed random number to anticipated

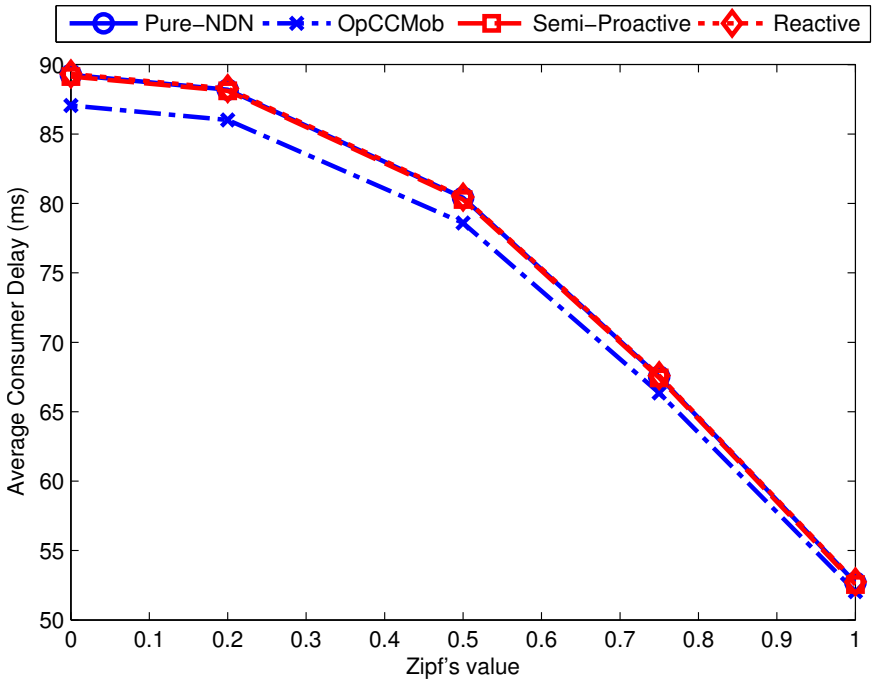



Figure 4.7: *OpCCMob* compared to other schemes under different data popularities, showing average Consumer delay at 50% mobility.

Table 4.4: Outcomes of mobility Prediction errors on *OpCCMob* and Semi-proactive schemes.

| | | Prediction | | |
|---|---|---|---|---|
| | | **HO** | | **No HO** |
| Actual | **HO** | Correct PoA | Wrong PoA | |
| | | Case 1 | Case 2 | Case 3 |
| | **No HO** | Case 4 | | Case 5 |

| | |
|---|---|
| Case 1 & 5 | Perfect prediction |
| Case 2 | Semi-proactive fails |
| Case 3 | Both schemes fails |
| Case 4 | Extra overhead |

Consumer's positions such that the PoA of nodes on the edge of an AP is altered to one of the neighboring PoA. This can lead to 5 outcomes summarized in Table 4.4.

When the prediction is correct (i.e., Cases 1 and 5), the schemes work as explained above with full knowledge of future handover events. For Case 4, when false positive occurs, the schemes try to optimize unnecessary requests, hence excess overhead is created on the network. The worst scenario is in Case 3, where a handover event is missed, since the schemes do not handle those *interests*, and retransmissions are required as in Pure-NDN. Finally, the case when the handover event is predicted but with wrong new PoA (Case 2), only the performance of the Semi-Proactive scheme is affected. This is because *OpCCMob* handles the handover before it occurs, therefore the exact information of the new PoA is not needed to optimally cache the data.

Figures 4.8 and 4.9 depict the delay and overhead of Semi-Proactive and *OpCC-Mob* schemes with error in mobility prediction. The average Consumer delay in the Semi-Proactive scheme increases by a small percentage to reach the upper bound (i.e., pure-NDN), whereas the delay in *OpCCMob* is increased by 75% when all location predictions are wrong. The extra delay is due to the scenarios of Case 3 mentioned above when handover events are missed. Nevertheless, there is still a performance gap for other schemes to improve. The overhead of both schemes is increased by 59%, due to Case 4 scenarios. The performance of the schemes can be improved by considering multiple future PoAs to avoid the uncertainty, which requires optimizing the overhead and caching resources.

### Imperfect Requests Prediction

*OpCCMob* uses request predictors to identify the data that should be pre-fetched before handover. If wrong requests are predicted, the delay of these requests will be optimized whereas the mobile requests will not. In this experiment, a uniform

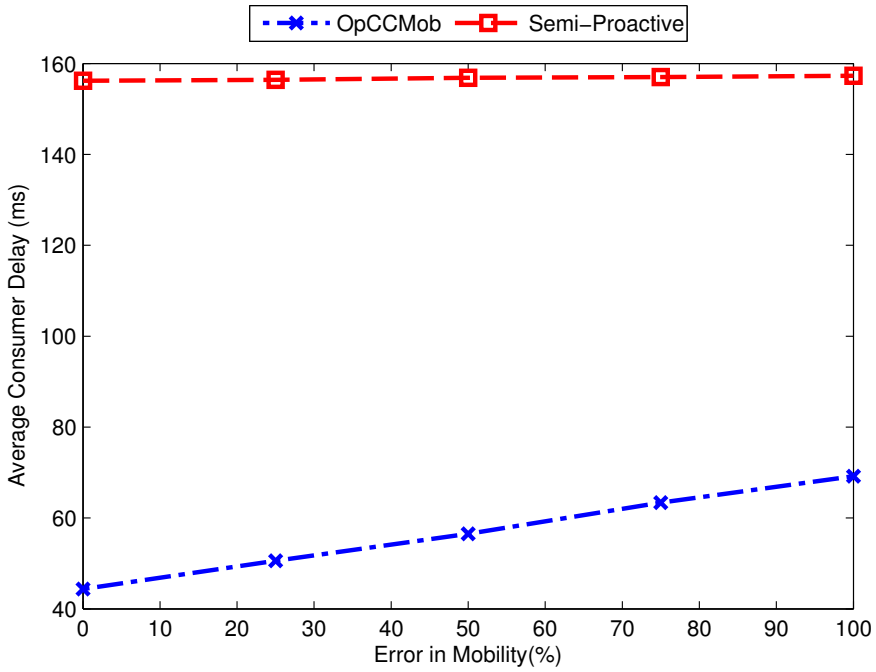



Figure 4.8: *OpCCMob* vs. Semi-proactive under different error in mobility percentages, showing average Consumer delay at 50% mobility.

random error is added to the predicted *interests* such that the name of the request is changed. Figure 4.10 demonstrates the average delay at 50% mobile Consumers for the Semi-proactive scheme and *OpCCMob* with 0%, 50% and 100% percentages of error in prediction. Furthermore, 3 different cache sizes are tested. The delay increases by 100% when 50% of the predicted requests are erroneous. Additionally, the performance is close to Semi-Proactive scheme when the predictor provides erroneous information.

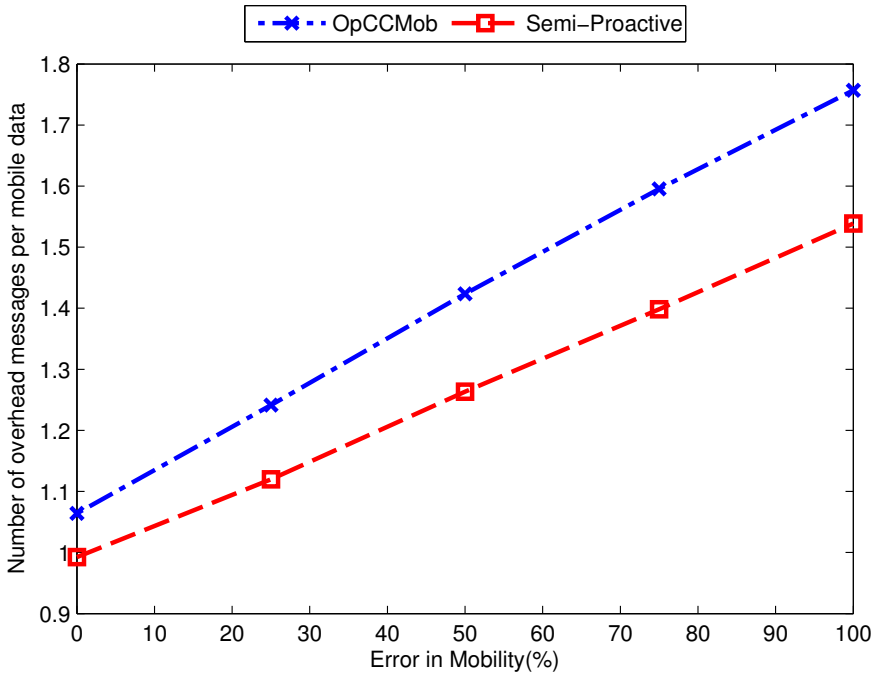



Figure 4.9: *OpCCMob* vs. Semi-proactive under different error in mobility prediction percentages, showing overhead per interest at 50% mobility.

## 4.5 Conclusions

We propose an optimal proactive Consumer mobility management scheme (*OpCCMob*) that uses in-network caching and location prediction to support seamless operations in NDN. The evaluation of *OpCCMob* shows the advantages of proactively caching the data before handover. In particular, the average delay is not changed with the increase of mobile events, however overhead packets are required. The results contribute as a benchmark for other schemes proposed currently or in future literature.

The performance of *OpCCMob* compared to the two main approaches of Consumer mobility (Reactive and Semi-proactive) provides a gap that can be filled by utilizing more cache resources. That is in addition to the use of mobility predictions to provide

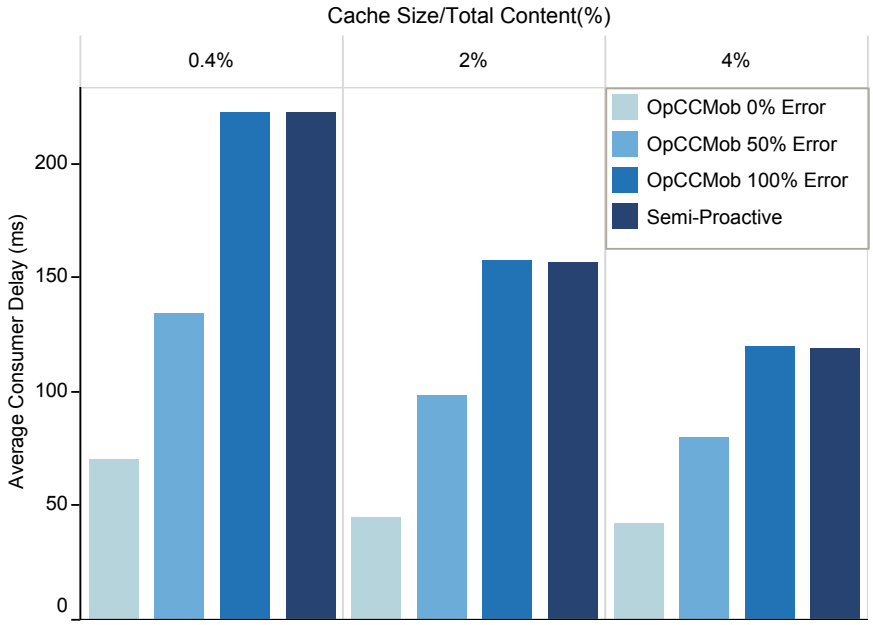



Figure 4.10: *OpCCMob* vs. Semi-proactive under different error in requests prediction percentages, showing average Consumer delay at 50% mobility.

fully proactive solutions. On the other hand, the overhead associated with *OpCCMob* is an upper-bound to other schemes. With the provided bounds on performance, a trade-off between an acceptable delay and overhead cost can be introduced. The sensitivity of *OpCCMob* to errors in location predictions and requests forecast has been analyzed. The results have shown that the proposed scheme is robust to imperfect location predictions and provides sub-optimal results to data requests' errors.

# Chapter 5

# Proactive Producer Mobility Solution

Several efforts have addressed the Producer mobility challenge in ICN generally, and NDN specifically. Whereas most of the proposed schemes use reactive techniques to recover after a mobile event, in this chapter, we propose a predictive caching framework to support seamless Producer mobility. Similar to the proactive Consumer mobility scheme proposed in Chapter 4, the framework exploits position prediction techniques in addition to predicting users' access patterns to proactively store potential data on in-network caches. Thus, *interests* generated during a Producer mobility event can be satisfied by the intentionally placed data items. Finding the best placement that guarantees no *interest* drops with bounded overhead has the potential to provide an optimal proactive solution to the Producer mobility problem.

First, we propose a design for this optimal scheme, called *OpProMob*. Second, we propose a polynomial time algorithm, *DCacheMob*, that provides a near-optimal solution in real-time by greedily finding the optimal placement of each *interest* separately. We use the assessment framework proposed in Chapter 3 to evaluate the effectiveness of the proposed schemes. This is in addition to measuring the gap in performance between the optimal caching approach and other mobility schemes.

The remainder of this chapter is organized as follows. In Sections 5.1 and 5.2, the optimal proactive benchmark and its formulation are explained. The practicality of *OpProMob* is discussed in Section 5.3. The real-time scheme *DCacheMob* is proposed in Section 5.4. The simulation experiments are discussed in Section 5.5. We conclude our findings in Section 5.6.

## 5.1 Proactive Caching Framework

The problem in Producer mobility is that during the convergence time, the network is not updated with the new location of the Producer. Hence, any *interest* that is directed to this Producer's old PoA will be dropped, unless it finds the *data* in one of the caches along the path from the Consumer to the Producer. The basic idea of *OpProMob* is to proactively place the data that will be requested during the convergence time in the caches before the handover occurs. Using predicted mobility events as well as the request patterns, we can then find the potential *interests* to be dropped after a Producer mobility event, and take the necessary action in advance to cache the data. We assume such information to be available and correct (i.e., perfect knowledge of the future), to find an optimal solution to the data caching problem.

Figure 5.1 shows the required information and the main blocks that interact with *OpProMob*. Specifically, the location predictor provides the future positions of each Producer. Moreover, the access profiles of all Producers which state whether users can connect to the nearest PoA or not. Given those two inputs and the network topology, the user-to-PoA association can be estimated. Furthermore, the requests predictor provides the request patterns of Consumers in the future. Specifically, it supples the information of when each Consumer will request data.

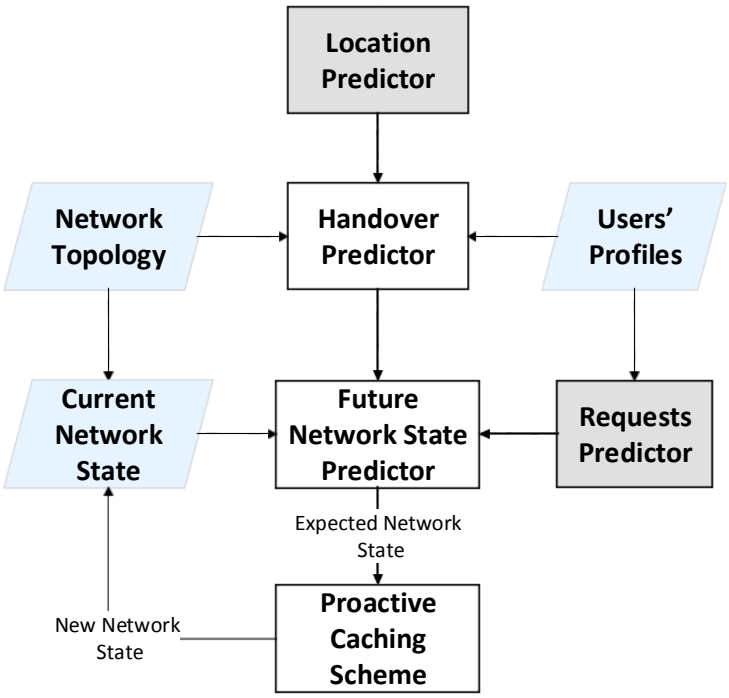



Figure 5.1: Proposed proactive framework. The Blue blocks are information provided to the framework, whereas Grey blocks are predictors that provide future knowledge. Proactive Caching scheme is the block that handles mobility.

The future state of the network is estimated by using the current state, predicted requests' patterns and user-to-AP association. The estimated state of the network is then fed to the proactive cache scheme, *OpProMob*, and the generated caching solution is applied back to the NDN.

The location predictor

Assume the handover event starts at $t_{HO}$ and that it takes $\tau$ seconds to finish, which includes the time needed by the Producer to connect to the new AP and the convergence time needed by the network to update the routing state. Moreover, $T$ is the average time to reach the Producer in its old PoA. Hence, *interests* directed to a mobile Producer and requested between $t_{HO} - T$ and $t_{HO} + \tau$ are predicted to be

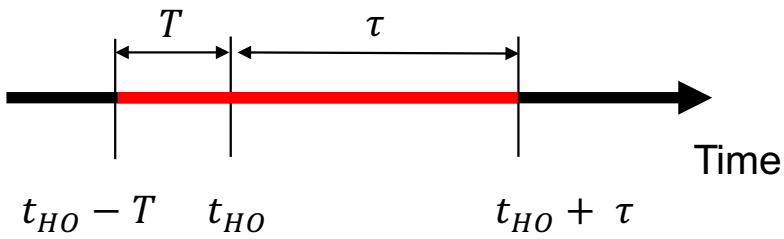



Figure 5.2: Producer mobility timeline. *Interests* requested within $[t_{HO} - T, t_{HO} + \tau]$ will be dropped.

dropped and will be retransmitted, see Figure 5.2. Given this set of requests, we can define the problem as finding the optimal placement of data in network caches such that no retransmissions of *interests* are required by the Consumer while maintaining bounded overhead on the network.

As mentioned earlier, caching the *data* required in advance allows the *interests* to be satisfied on the first attempt. However, this comes with an overhead which can be of three types:

1. Path updates: Data can be placed on any cache in the network. Caching on a router that is not on the path from a Consumer to a Producer requires a route to be created temporarily from the Consumer to the router chosen. Hence, an overhead of updating the FIBs of the new path is incurred.

2. Cache cost: Adding *data* to a full cache requires replacing some other items, based on the replacement policy in place. Both the total number and value of *data* removed should be bounded to avoid affecting other users.

3. Producer traffic: The mobile Producer will be required to handle the proactive requests before the handover in addition to its regular load. This traffic is an

overhead on the Producer and should be bounded.

## 5.2 Problem Formulation

Using the same system model explained in Section 4.2, finding the best placement of the set of *data* with bounded overhead can be formulated as an optimization problem as follows:

$$\min_{\delta_r^d, \rho_r^m} \sum_{d \in D} \sum_{r \in R} |P_{u(d) \to r}| \delta_r^d \tag{5.1}$$

S.t.

$$\sum_{r \in R} \delta_r^d = 1 \quad \forall d \in D \tag{5.2}$$

$$\sum_{d \in D} \sum_{r \in R} |P_{u'(d) \to r}| \delta_r^d < \zeta \tag{5.3}$$

$$\sum_{d \in D} \sum_{r \in R} \mathbb{P}_d^r < \sigma \tag{5.4}$$

$$\sum_{m=0}^{c_r} \rho_r^m = 1 \quad \forall r \in R \tag{5.5}$$

$$c_r + \sum_{d \in D} \delta_r^d - \sum_{m=0}^{c_r} m \rho_r^m \leq c_r^{max} \quad \forall r \in R \tag{5.6}$$

$$\sum_{d \in D} \delta_r^d - \sum_{m=0}^{c_r} m \rho_r^m \geq 0 \quad \forall r \in R \tag{5.7}$$

$$\sum_{m=0}^{c_r} m \rho_r^m < \alpha \quad \forall r \in R \tag{5.8}$$

$$(\sum_{m=0}^{c_r} \rho_r^m k_r^m - \sum_{d \in D} \epsilon_d . \delta_r^d) / K_r < \omega \quad \forall r \in R \tag{5.9}$$

$$c_r^{Sp} + \sum_{d \in D} \delta_r^d \leq c_r^{SpMax} \quad \forall r \in R \tag{5.10}$$

The decision variables of the optimization problem are:

$$\delta_r^d = \begin{cases} 1 & \text{Place } \textit{data } d \text{ on router } r \\ 0 & \text{Otherwise} \end{cases}$$

$$\rho_r^m = \begin{cases} 1 & \text{Remove } m \text{ items from router } r \\ 0 & \text{Otherwise} \end{cases}$$

The objective function (5.1) is to minimize the delay experienced by the Consumer through minimizing the number of hops the *interests* travel.

Using similar equations from Section 4.3, Constraints (5.2) to (5.10) are of 3 types; 1) constraints to bound the overhead 2) constraints to avoid violating caching capacities and 3) constraints to bound the range of the decision variables. In more details, Constraint (5.2) ensures that all *data* will be cached once in any of the routers, whereas Constraint (5.3) bounds the amount of traffic generated to support mobility (the third type of overhead). Specifically, the summation of the number of hops the *data* has to travel should be less than a threshold $\zeta$. Constraint (5.4) bounds the total number of path updates. In particular, the number of path updates for *data* $d$, $\mathbb{P}_d^r$, as defined in Section 4.3, is the number of non-common edges between $P_{u(d)\rightarrow u'(d)}$ (which is the path from Consumer $u(d)$ to Producer $u'(d)$) and $P_{u(d)\rightarrow r}$ (which is the new path from Consumer $u(d)$ to chosen router $r$).

Constraints (5.5) to (5.10) are related to caching and *data* replacement. Constraint (5.5) is used to force caching each data exactly once. To ensure that the number of

items in the cache does not exceed the max capacity, (5.6) is used. Constraint (5.7) ensures that unnecessary replacements are avoided (i.e., does not remove more than data added). To bound the number of replacements per router, (5.8) is used with a threshold $\alpha$. Since some content is replaced, the total content value of the router changes. The reduction of this value is bounded by a threshold $\omega$ in Constraint (5.9). The value of variable $k_r^m$ is the total content value of $m$ items to be replaced from router $r$, and $K_r$ is the total content value of all the items in $r$.

The proactive part of the cache design discussed in Section 4.1.2 can be expanded to store more special *data* as long as it does not exceed the maximum capacity allocated to it ($C_r^{SPMax}$). Since the data in the special cache should not be replaced, the amount of *data* to be stored in a router should not exceed the number of empty slots in the special cache. Constraint (5.10) enforces this condition, where $C_r^{SP}$ is the current number of items in the special cache.

The formulation is considered 0-1 ILP, since all equations are linear and the decision variables are binary. As in previous formulation in Chapter 4, an optimal solution can be obtained using commercial solvers.

## 5.3 Optimal Benchmark vs. Real-time Solutions

The *OpProMob* formulation guarantees seamless operations during mobility events by providing the minimal Consumer delay for *data* during the Producer handover. In this section, we address the solution methods for such a formulation and highlight their main advantages and limitations.

Existing mathematical optimization techniques and commercial solvers can provide a global optimal solution for the formulation in hand. Simplex, branch and

bound are among these techniques; however, the following hinders usage in real-time:

1. High Complexity:

   The time needed to get the solution should be minimal in order to provide seamless mobility. However, *OpProMob* is NP-hard and to obtain the optimal solution for the 0-1 ILP formulation, long computations are required. Therefore, a near-optimal solution that can be implemented in real-time with low complexity is required.

2. Centralized:

   *OpProMob* requires information about the whole network such as the current topology and performance statistics. To get this information *OpProMob* needs to be centralized in one unit to collect the required information and then execute the solution. Having one unit in a large network poses a scalability problem and adds a risk of single point of failure.

3. Sensitivity to errors:

   Since the scheme uses information—from predictors—which is bound to have errors [32], the results are affected.

To summarize, the complexity of finding the optimal *data* placement is intolerable for real-time decision making. Therefore, optimal solvers and mathematical methods can be used only to provide benchmark solutions and evaluate other mobility management proposals. Next, we propose a scheme that can find a decentralized near-optimal solution with lower complexity for the proposed predictive Producer based mobility caching and we leave the 3rd problem to Chapter 6.

## 5.4 Decentralized Real-time and Near-Optimal caching

In this section, we propose a practical scheme that consider the problems of *Op-CacheMob*. The proposed scheme, denoted by *DCacheMob*, uses guided heuristics to find a solution to the predictive caching problem in a decentralized way and with low complexity to be used in real-time. *DCacheMob* runs separately for each domain of the network where the input and decisions are taken locally for the local Consumers in the domain. In particular, it predicts local Producers mobility events and local Consumers requests and then finds the solution to be applied on local routers. Hence, *intra-domain* traffic is considered in this case. On the other hand, *inter-domain* traffic where local Consumers request *data* from remote Producers requires extra information about the mobility events of the remote Producers. This can be provided by pushing predicted handover events of local Producers to all other entities running *DCacheMob*, such that the set of predicted *interests* of Intra and Inter-domain traffic is found. The framework proposed in Figure 5.1 is modified to make it run in a decentralized fashion. In particular, the set of global handover events is an additional information block needed by the location predictor. Moreover, all other information is now locally gathered in each domain. The proactive scheme uses this information to find the optimal placement in each domain without the need for a central unit.

In each domain, *DCacheMob* finds the optimal placement for each *interest* individually. Moreover, the input set is sorted by request or handover time whichever is sooner. This is to give priority to *interests* that will be dropped first if their *data* is not cached. For each *interest*, the router with the least amount of path updates (Equation (5.1)) is selected to cache the corresponding *data*. The router has to be feasible, in which constraints (5.2) to (5.10) should be satisfied for the selected router.

**Algorithm 1** *GetNetworkMetrics*: Function to extract metrics from the current state of the network.

**Input:** *network*: The current state of the network
**Input:** $D$: Set of requests to be optimized
**Output:** $PathUpdates[D][R]$: 2D array to hold the number of path updates for each pair $(d,r)$
**Output:** $hops[D][R]$: 2D array to hold the number of hops to reach $r$ from Consumer of $d$
**Output:** $\alpha$: vector of size $D$ to hold $\gamma$ thresholds of each $d$
**Output:** $\zeta$: variable to hold the threshold $\zeta$

```
1:  R ← getSetOfRouters(network)
2:  for all d ∈ D do
3:      cons ← GetConsumer(d)
4:      prod ← GetProducer(d)
5:      p1 ← getPath(network,cons,prod)
6:      γ[d] ← |p1|
7:      ζ ← ζ + γ
8:      for all r ∈ R do
9:          p2 ← getPath(network,r, prod)
10:         pathUpdates[d][r] ← getNoncommon(p1,p2)
11:         hops[d][r] ← getNumofHops(cons,r)
12:     end for
13: end for
```

If it is not feasible, the next available router will be chosen. If no feasible router is found, this interest will be ignored, and the next one is taken. This greedy approach does not guarantee an optimal solution, but it provides a near-optimal one. This is because the decision considers the main objective which is delay minimization while satisfying all the constraints.

The *DCacheMob* consists of the following four main stages:

1. Data Preparation:

   The function in Algorithm 1 is used to collect statistics from the network before taking decisions. In particular, for every pair of *interests* and routers, the

**Algorithm 2** Pseudo-code of *DCacheMob*.

**Input:** *network*: The current state of the network
**Input:** $D$: Set of requests to be optimized

```
1:  Initialize Sol.DataAdded[r] = 0 ∀r ∈ R
2:  Initialize Sol.DataRemoved[r] = 0 ∀r ∈ R
3:  GetNetworkMetrics(network,D,R)
4:  for all d ∈ D do
5:      R' ← SortRouters(d, R)
6:      for all r ∈ R' do
7:          Sol' ← Sol
8:          Sol'.DataAdded[r] ← Sol'.DataAdded[r] + 1
9:          if C_r + Sol.DataAdded[r] − Sol.DataRemoved[r] = C_r^max then
10:             Sol'.DataRemoved[r] ← Sol'.DataRemoved[r] + 1
11:         end if
12:         if Sol'.isFeasible(d,r) then
13:             Sol ← Sol'
14:             Sol.SelectedRouter[d] ← r
15:             Break
16:         end if
17:     end for
18: end for
19: ApplySolution(Sol)
```

number of path update messages needed to reach the router and the number of hops to the Consumer is calculated.

2. Greedy Placement and Removal:

   The greedy algorithm, in Algorithm 2, iterates over the set of *interests* that are predicted to be dropped (line 4). For each *interest* $i$, the algorithm firstly sorts (line 5) the full list of caching routers in ascending order based on the number of path updates needed to direct *interest* $i$ to the routers. This aims to minimize the network overhead. Then, the algorithm (in line 6) sequentially selects the routers and tests the possibility of caching the *data* of *interest* $i$. In the case of

capacity constraint violation, *data* replacement occurs (lines 9-11).

3. Feasibility Check:

   The feasibility of the above *data* placement and removal actions are then checked using Algorithm 3 which is called by Algorithm 2 at line 12. In particular, the solution (i.e., selected router) that satisfies all constraints in (5.2) to (5.10) is considered as optimal and the *data* of *interest* $i$ will be stored in the current router $r$. Otherwise, if any of the constraints are violated (i.e., Algorithm 3 returns false), the solution is said to be infeasible and the next router will be checked. In the case of visiting all the routers without finding a feasible one, the *data* of *interest* $i$ will not be cached and will be sent upon the user's request.

4. Solution Broadcasting:

   After selecting the *data* of all *interests*, the last stage places the *data* on the chosen routers and send FIB update messages to the off-path routers.

### 5.4.1 Complexity Analysis

To obtain the optimal solution, the optimizer needs to solve 0-1 ILP problem. Given the formulation in Section 5.2, the number of decision variables is $R(D+c_r^{max})$ and the number of constraints is $2D+6R$. On the other hand, the complexity of *DCacheMob* is polynomial. Specifically, the algorithm loops on the set of *data* and sorts the list of routers per iteration. The sorting algorithm used to sort routers has a complexity $O(RLog(R))$, then the complexity of the algorithm is $O(DRLog(R))$.

**Algorithm 3** Data structure *Solution* with attributes and feasibility test function.

**class** *Solution*
+ $PathUpdates$ ▷ Total path updates needed to apply the solution
+ $ProducerHops$ ▷ Total number of hops needed to send all *data*
+ $DataAdded[R]$ ▷ Array to hold number of added *data* per router
+ $DataRemoved[R]$ ▷ Array to hold number of removed *data* per router
+ $ValueDataAdded[R]$ ▷ Array to hold the value of added *data* per router
+ $ValueDataRemoved[R]$ ▷ Array to hold the value of removed *data* per router
+ $SelectedRouter[D]$ ▷ Arrays to hold the selected router of each *data*

```
function isFeasible(d,r)
    cons ← GetConsumer(d)
    prod ← GetProducer(d)
    P1 ← getPath(network,prod,r)
    P2 ← getPath(network,cons,r)
    if ProducerHops + |P1|> ζ or |P2|> γ then
        return False
    end if
    if PathUpdates + PathUpdates[d][r] > ω then
        return False
    end if
    if C_r + DataAdded[r] − DataRemoved[r] > C_r^max then
        return False
    end if
    if DataAdded[r] − DataRemoved[r] > 0 or DataRemoved[r] > α then
        return False
    end if
    if ValueDataRemoved[r] − ValueDataAdded[r])/k_r > ω then
        return False
    end if
    if C_r^Sp + DataAdded[r] > C_r^SPmax then
        return False
    end if
    ProducerHops ← ProduerHops + |P1|
    PathUpdates ← PathUpdates + PathUpdates[d][r]
    return True
end function
end class
```

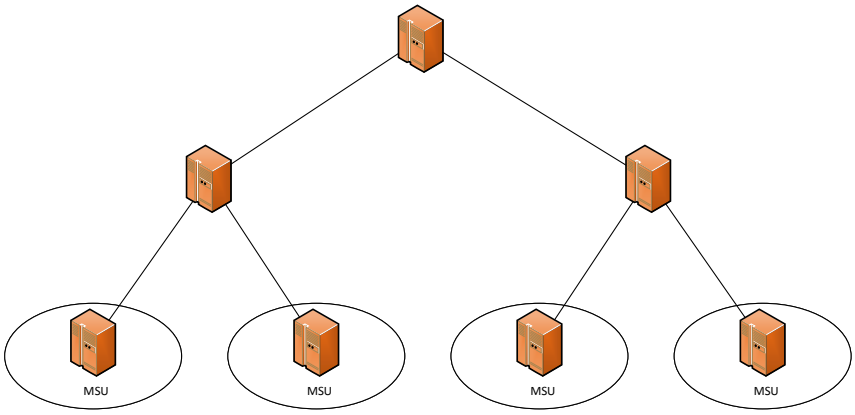



Figure 5.3: A tree topology example to distribute handover events using MSUs.

### 5.4.2 Practical Utility

This section explains how the proposed scheme can be practically implemented in NDN. First, an entity called Mobility Support Unit (MSU) is installed in each domain of the network to predict local information and take local actions. Additionally, each MSU is connected to a tree of units that is used to gather Producer movements. In particular, the entities are leaves in the tree and each node in the tree is aware of all Producer movements occurring in the descendant's nodes, as shown in Figure 5.3. This decentralization of responsibilities adds scalability to the scheme to support large networks. Alternatively, the tree can be constructed within the topology by find the Minimum Spanning Tree (MST) of the MSUs of all domains.

In more details, each MSU is responsible for the following:

1. Predict local Producer handover events.

2. Upload mobile predictions to the upper level.

3. Predict local Consumers request patters.

4. Download mobile predictions of remote Producers from the upper level.

5. Using information from 1 & 2, run *DCacheMob* to find a near-optimal solution.

6. Apply the solution by sending action commands.

The action commands are sent to routers in *interest* packets by encapsulating the action to be taken in the name field Specifically, there are two types of actions; update FIB entry or request *data*. The first type asks the router to add a new FIB entry for a specific *data* such that it is forwarded to the cached copy instead of the Producer. The second type makes the router request a copy of the *data* from the Producer and cache it until the Consumer asks for it. Moreover, the *interests* are forwarded to all routers involved in the decision taken by the entity are decoded to extract the actions which are then applied. Hence, a new logic should be added to the routers such that these actions can be processed and acknowledgment packets can be sent back to the entity.

The steps taken after deciding on the placement can be summarized as follows:

1. MSU sends an *interest* packet that includes the *data* name required to be requested and the time that it should be requested at. Moreover, it includes the FIB entry to be added to the router, if any.

2. Upon receiving control packets, the router schedules sending *interests* with the *data* names at the times mentioned in the commands.

3. The Producer will send the *data* requested with no changes to the regular process.

4. The router will cache the *data* received waiting for Consumer's requests.

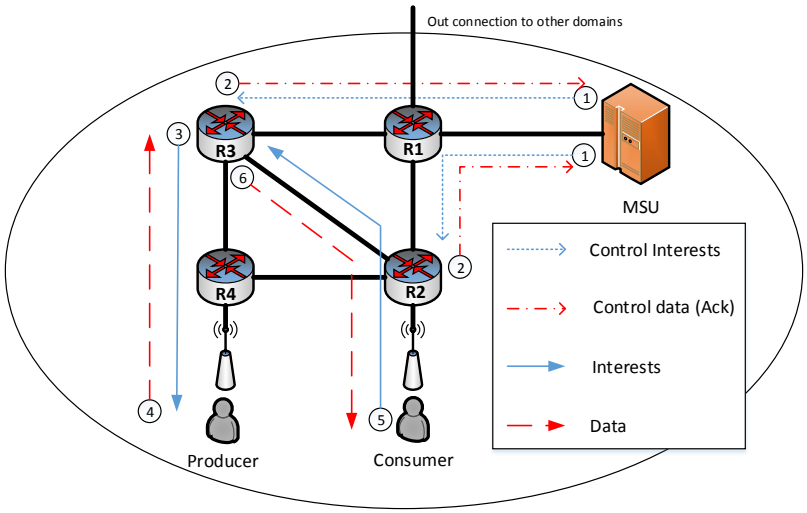



Figure 5.4: An example that shows the process of content placement in *DCacheMob*.

The steps can be shown in Figures 5.4 and 5.5. In particular, Figure 5.4 illustrates an example where MSUs are installed in 4 domains. In domain 1, the Consumer is predicted to request a *data* from a Producer that moves to a different PoA. The entity takes a decision before the handover to the cache the *data* in router R3. To be able to complete the action, Router R2 should forward the *interest* to R3 instead of the Producer. Hence, the entity will send control *interests* to R2 and R3. At the time assigned by the entity, Router R3 then sends an *interest* to the Producer asking for the *data* to be cached. Finally, the *data* is cached in R3 and waiting for the Consumer to send the *interests* normally which will be forwarded to R3 since R2 has an FIB entry for this *data*. Figure 5.5 shows the timeline of the above mentioned events.

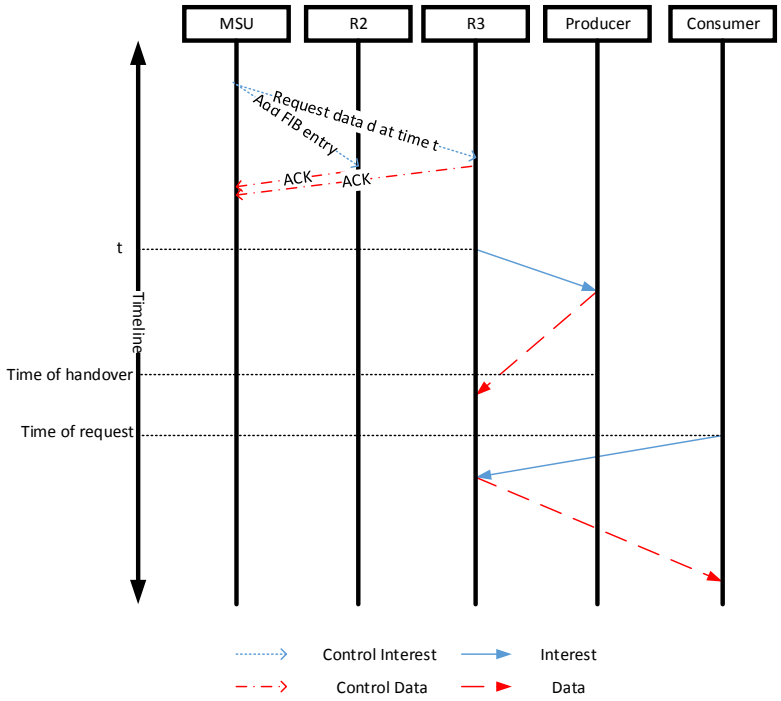



Figure 5.5: Timeline of *interests* and *data* sent between all entities in *DCacheMob*.

## 5.5 Results and Discussion

### 5.5.1 Implementation of Proactive Schemes in the Assessment Framework

The optimal scheme *OpProMob* is implemented in the proposed assessment framework of Chapter 3. The implementation is similar to the one presented in Section 4.4.1 where the formulation is modeled in Gurobi and the solution found is then applied to the network. In the case of *DCacheMob*, the MSU is chosen as a node in each domain which runs the greedy algorithm at the start of every optimization window. The solution resulted from the heuristic approach is then applied to the local domain as in *OpProMob*.

Table 5.1: Simulation thresholds of *OpProMob* and *DCacheMob*.

| Parameter | Value |
|---|---|
| $\alpha$ | 10% |
| $\omega$ | 10% |
| $\sigma$ | 20% |
| $\zeta$ | Average number of hops |
| Optimization window | 2 seconds |

### 5.5.2 Experimental Setup

The experimental setup used is explained in Section 3.4. Additionally, the thresholds of the proposed schemes are summarized in Table 5.1.

### 5.5.3 Comparative Schemes

We evaluate both schemes *OpProMob* and *DCacheMob* by comparing them to three mobility management approaches. Namely, NDN with no scheme (Pure-NDN), Mobility Anchor (*MA*) represented in [62] and Location Resolution Scheme (*LRS*) represented in [57].

### 5.5.4 Producer Mobility

The first scenario evaluates the aforementioned mobility management schemes under different number of mobile Producers in the network. The results in Figure 5.6 and Figure 5.7 demonstrate the ability of the introduced schemes to proactively cache the future content to be requested. Therefore, they avoid retransmission of *interests* to the new location and attain the maximum delivery ratio irrespective of the mobile Producers percentage. Moreover, the optimal placement of the content near the Consumer's location results in reduced delay, which is also stable over the percentage

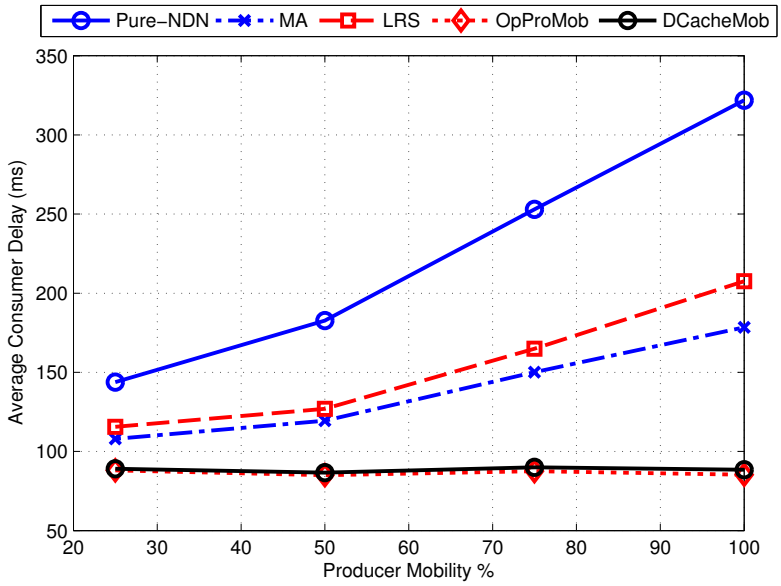



Figure 5.6: *OpProMob* and *DCacheMob* compared to reactive schemes under different mobility percentages, showing average Consumer delay over all *interests*.

of mobility events as shown in Figure 5.6. These optimality conditions require extra overhead, shown in Figure 5.8, for delivering the mobile Producer's content to the selected cache and sending path update packets in the network.

As such, the results reveal a significant deterioration of the total delay which increased by more than 100%, in the case of the pure-NDN, when only 50% of the Producers change their location. This is attributed to the simplicity of NDN that depends on *interest* retransmissions to recover from Producer Mobility and thus it suffers from a poor delivery ratio as shown in Figure 5.7, especially at a higher percentage of mobile Producers.

The existing non-predictive *MA* and *LRS* attain acceptable delays at low Producers mobility (i.e., 25%), redirecting the *interests* to the new locations is suboptimal at more dynamic scenarios. Thus, when 75% of Producers are mobile, the Consumers

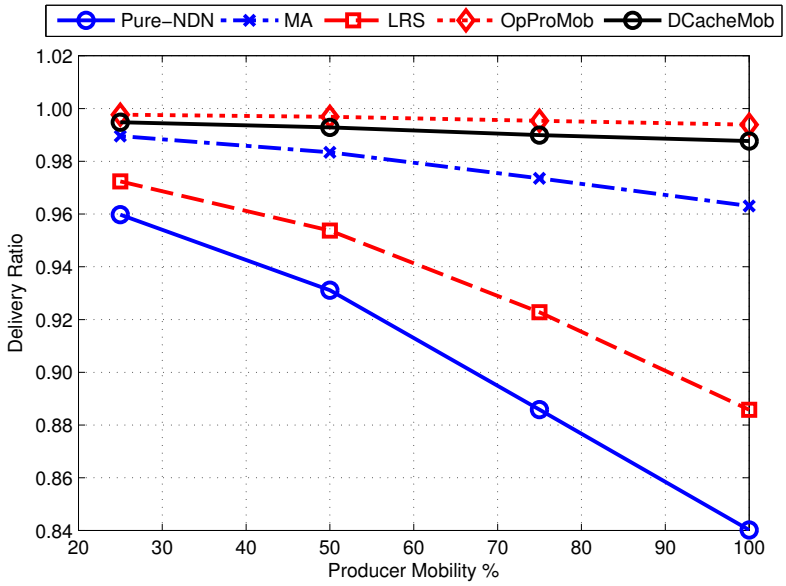



Figure 5.7: *OpProMob* and *DCacheMob* compared to reactive schemes under different mobility percentages, showing the delivery ratio over.

suffer from a 90% and 65% increase in the delay compared to the lower bound, predictive schemes, using *LRS* and *MA*, respectively, as shown in Figure 5.6. This delay is also associated with an increased number of *interest* retransmissions when the original *interests* become outdated for the Consumers (i.e., after the time out). Thus, a drop in the delivery ratio is also experienced by the network under these non-predictive techniques as shown in Figure 5.7. Figure 5.9 depicts more detailed results by showing the average Consumer delay for the *interests* that were sent during handover only.

The performance of the heuristic approach *DCacheMob* is also shown in Figures 5.6 to 5.9. In particular, the scheme achieves a very close performance to the optimal solution *OpProMob*. Moreover, it achieves such a performance with lower complexity as explained before in Section 5.4.

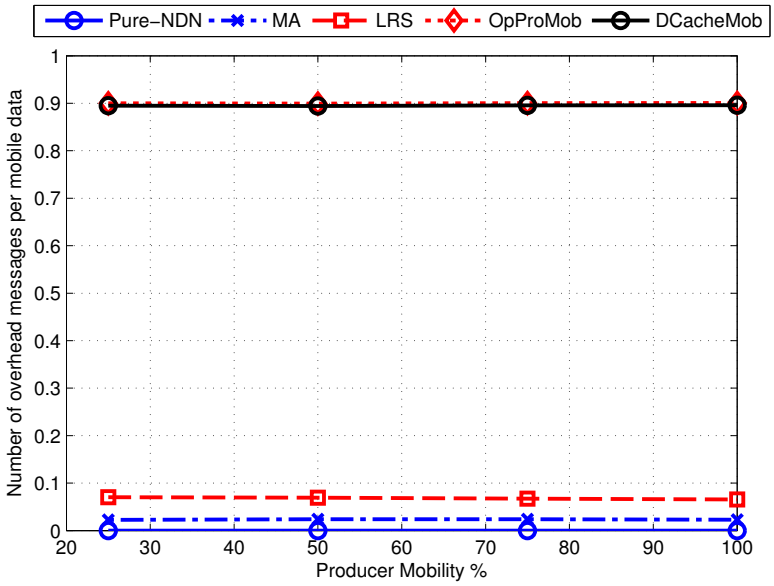



Figure 5.8: *OpProMob* and *DCacheMob* compared to reactive schemes under different mobility percentages, showing the overhead percentage.

While the predictive schemes evaluated the pure-NDN, *MA* and *LRS*, the overhead gap between the optimal scheme and each of these schemes provides room for enhancing their performance. In particular, these non-predictive mobility schemes can utilize the available network capacity (i.e., increase the overhead) in order to decrease the delay and delivery ratio gap.

The hit ratio in the non-predictive schemes decreases by 55% when all Producers are mobile as shown in Figure 5.10. This is because *MA* and *LRS* depend on renaming *interests* to be redirected to the correct location which makes the *data* cached previously by its original name obsolete.

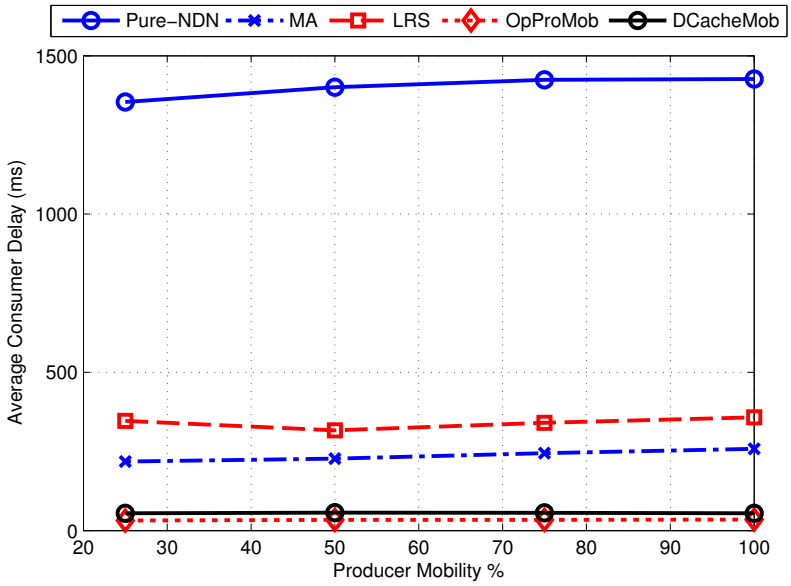



Figure 5.9: *OpProMob* and *DCacheMob* compared to reactive schemes under different mobility percentages, showing average Consumer delay for mobile *interests* only.

### 5.5.5 Cache Size

In this scenario, we test the ability of each scheme to exploit the available cache size and improve the Consumer's experience. This is done by changing the *Cache Percentage* which is the ratio between the Cache capacity and the total available content. The simulation is performed with 50% of the Producers are mobile.

The predictive schemes efficiently utilize any minor availability in the cache which significantly improves the Consumer delay, as shown in Figure 5.11. The degree of enhancement increases with the availability of more caching space where more *data* for the mobile Producers can be stored close to their requesting users. As such, the predictive schemes continue to provide a lower bound of delay compared to the existing non-predictive *MA* and *LRS*.

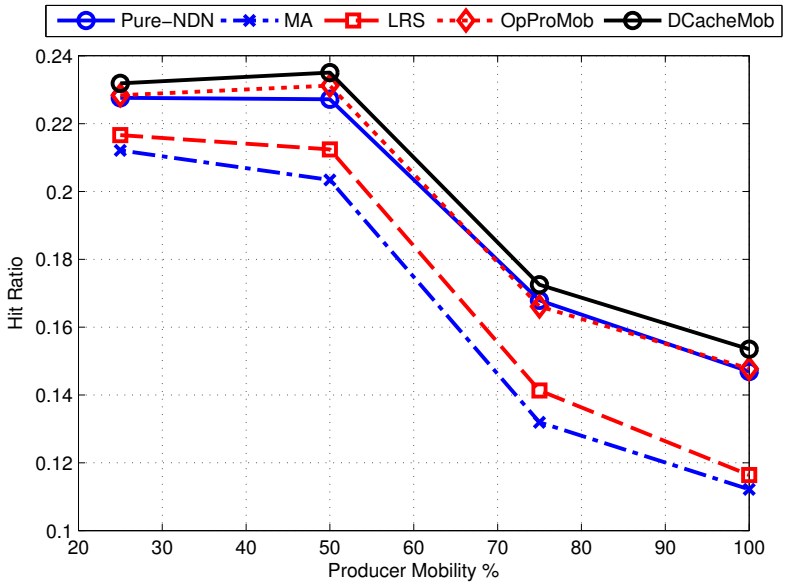



Figure 5.10: *OpProMob* and *DCacheMob* compared to reactive schemes under different mobility percentages, showing average hit ratio.

It is shown that *DCacheMob* fails to perform when the resources are limited. This suboptimal performance is the result of the local optimal decisions of each *interest*. Figure 5.12 illustrates that the amount of *OpProMob*'s overhead is 22% more in scenarios with low cache resources.

### 5.5.6 Sensitivity Analysis

The following experiments test the robustness of *OpProMob* and *DCacheMob* to errors in predictions.

#### Imperfect Mobility Prediction

Similar to the sensitivity analysis in Section 4.4.6, to Producer errors in prediction, we add a uniformly distributed random number to anticipated Producer's positions

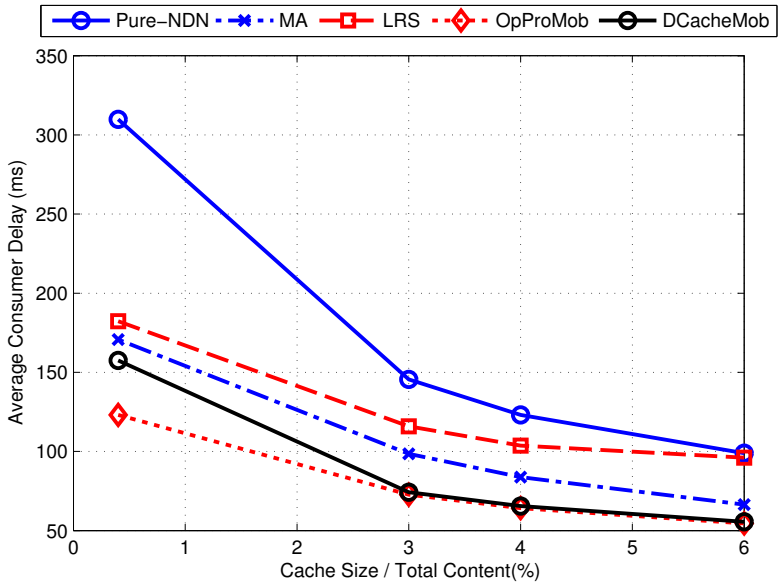



Figure 5.11: *OpProMob* and *DCacheMob* compared to reactive schemes under different cache size percentages, showing average Consumer delay over all *interests*.

such that the PoA may change to one of the neighboring PoA. In the same way, an error in estimating future location leads to one of the five cases summarized in Table 5.2, and the proposed schemes fail only in one of these outcomes. Specifically, the scheme performs perfectly in cases where the predictor estimates the correct PoA of a Producer, whether it is a handover or not (Cases 1 and 5, respectively). The false positive (Case 4), where wrong PoA is predicted, leads to false handover and causes the scheme to find the optimal placement for unrequested *data*. Consequently, extra overhead is generated but the Consumer is not directly impacted. The false negative (Case 3) is the only outcome that affects the scheme since *interests* directed to the moving Producer will not be considered. While Case 2 is a wrong prediction, it does not affect the scheme since the knowledge of future PoA is not needed.

Figure 5.13 and Figure 5.14 depict the delay and overhead of the two predictive

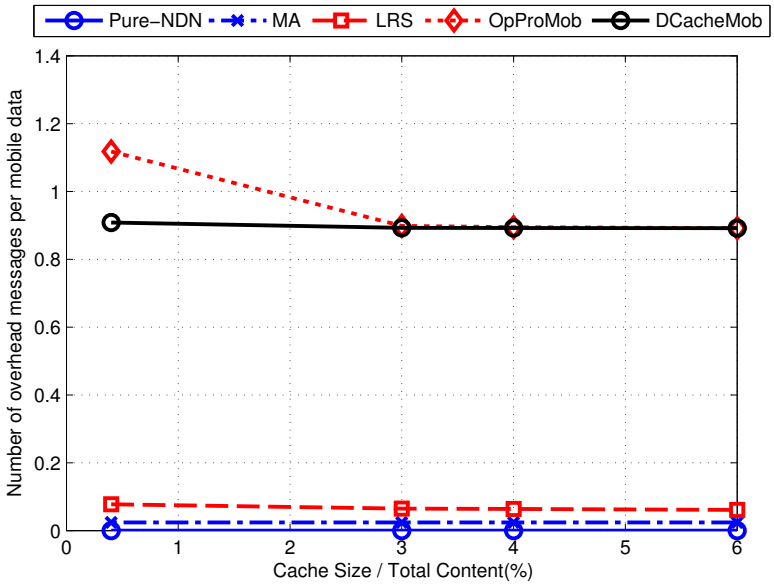



Figure 5.12: *OpProMob*, *DCacheMob* compared to reactive schemes under different cache size percentages, showing overhead percentages.

Table 5.2: Outcomes of mobility prediction errors on *OpProMob* and *DCacheMob* schemes.

| | | Prediction | | |
|---|---|---|---|---|
| | | **HO** | | **No HO** |
| Actual | **HO** | Correct PoA | Wrong PoA | |
| | | Case 1 | Case 2 | Case 3 |
| | **No HO** | Case 4 | | Case 5 |

| | |
|---|---|
| Case 1 & 5 | Perfect prediction |
| Case 2 | Wrong prediction, but no impact |
| Case 3 | The schemes fail |
| Case 4 | Extra overhead |

schemes with an error in mobility prediction. The average Consumer delay of all the schemes is increased by 9% when all locations are erroneous. The extra delay is

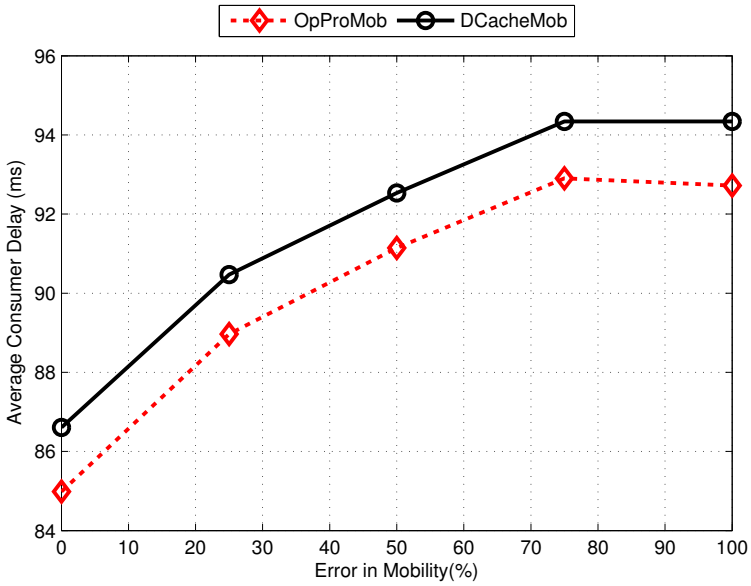



Figure 5.13: *OpProMob* and *DCacheMob* under different error in mobility prediction percentages, showing average Consumer delay over all *interests*.

due to the scenarios of Case 3 mentioned above when handover events are missed. Nevertheless, there is still a performance gap for other schemes to improve. For example, the gap between Mobility Anchor schemes and the predictive schemes while wrong predictions are made is 90%. On the other hand, the overhead is increased by 33% due to Case 4 scenarios. The performance of the schemes can be improved by considering multiple future PoAs to avoid the uncertainty, which requires optimizing the overhead and caching resources.

### Imperfect Consumer Request Prediction

The proactive schemes proposed use request predictors to identify the *data* that should be pre-fetched before Producer's handover. If wrong requests are predicted, the delays of these requests are optimized whereas the mobile requests are not. We design an

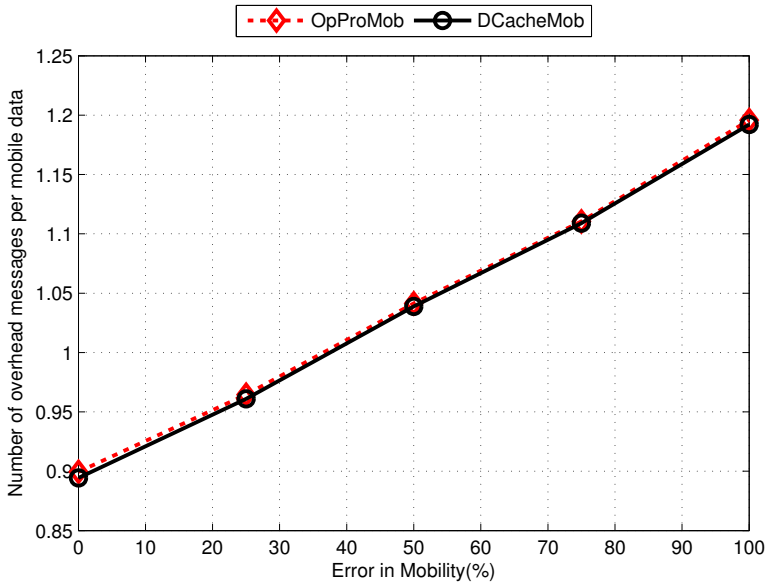



Figure 5.14: *OpProMob* and *DCacheMob* under different error in mobility prediction percentages, showing overhead percentages.

experiment such that a uniform random error is added to the predicted *interests* which changes the name of the request. Figure 5.15 demonstrates the average delay at 50% mobile Producers for the Mobility Anchor scheme and *DCacheMob* with 0%, 25% and 50% of error in prediction. Moreover, three different cache sizes are tested. The delay increases on average by 57% during 50% wrong predictions. This experiment shows that the proactive scheme fails when request errors exit and a robust scheme is needed.

## 5.6 Conclusions

This chapter presents our design and implementation of an optimal Producer mobility management scheme *OpProMob* that exploits the available in-network caching resources, and both location and data predictions. Moreover, a real-time solution

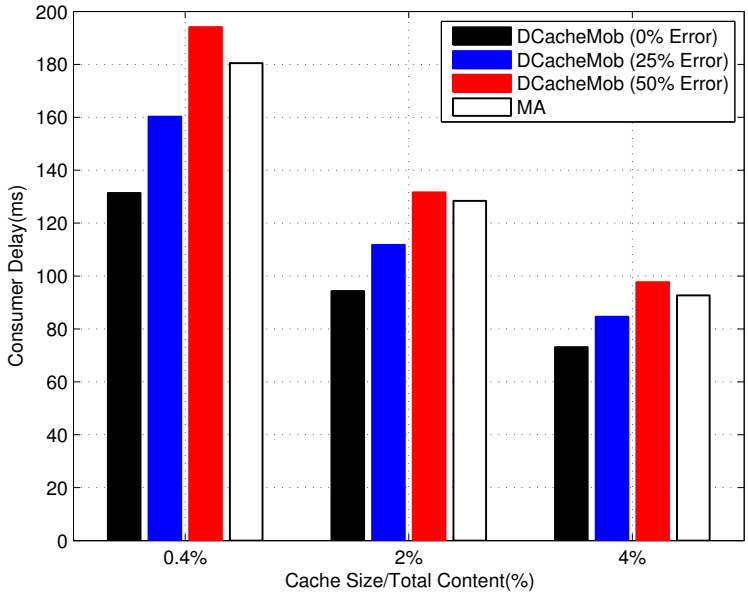



Figure 5.15: *DCacheMob* under different error in request prediction percentages, showing average Consumer delay.

*DCacheMob* is designed to give near-optimal results with lower complexity. Under perfect knowledge, the results demonstrated the ability of *OpProMob* and *DCacheMob* to proactively cache the future content of the anticipated mobile Producers. While compromising the network overhead, both the Consumer delay and delivery ratio remained almost constant, regardless of the number of mobility events, as long as there is enough space in the content store. Such results provide a *benchmark* for the main mobility solutions in the literature. This is in addition to provisioning insights on the trade-off between the prediction effort and overhead, on one hand, and the Consumer delay and delivery ratio on the other hand.

Nevertheless, the performance gap between the existing mobility management solutions and the *OpProMob* stresses the need to select the optimal cache size that guarantees the maximum tolerated delay. Such outcomes also provide a new trade-off

between the cost of increasing the cache size and the network overhead associated in the light of the predictive strategy.

Moreover, sensitivity analysis demonstrated the robustness of the introduced solutions to imperfect mobility predictions and suboptimal traffic forecasts. Thus, the analysis provides the network operator with accuracy requirements for the prediction techniques and the corresponding operational region of the predictive schemes.

# Chapter 6

# Robust Producer Mobility Framework

As shown in the previous chapter, one of the promising approaches to support seamless Producer mobility is proactive caching where network resources are used to store predicted future *interest* requests. The approach was evaluated in Section 5.5 where the objective was to minimize the delay and bound the overhead. Moreover, *idealistic* conditions with perfect prediction were assumed. Compared to conventional non-predictive schemes, such proactive management of Producer mobility results in satisfying all the delay requirements of Consumers.

In this Chapter, we propose a delay-sensitive benchmark and we support the practicality of proactive mobility management by considering uncertainty in predicted requests. Frequent changes in user demands or erroneous previous information typically result in imperfect prediction of future *interest* requests. Thus, the cached content is not going be used, resulting in suboptimal utilization of caching resources. This is in addition to compromising *consumers'* QoS due to missing the true content requested during Producer handover.

Previously in Chapter 5, we assumed that full knowledge of the future is given with no errors in predictions of mobile handover or requests. In this chapter, we propose a

proactive scheme that is robust to errors in predictions. Moreover, we focus on errors in request predictors since the proactive scheme is not sensitive to location errors due to the needed coarse level of prediction as discussed in Section 5.5.6.

Mainly, we introduce a *stochastic* optimization framework for proactive Producer mobility management that is *robust* to content prediction errors. In addition to caching the most probable future content, the framework adopts prediction errors model to cache other less probable content that might be requested by the users. Thus, it avoids an increased Consumers delay if such less probable content is not cached in advance. We formulate the Producer mobility management problem using Chance Constraint Programming (CCP). In particular, predicted *interest* requests are modeled as random variables in order to incorporate the associated uncertainty. The CCP bounds the satisfaction degree of network constraints by a certain probabilistic level [6]. Thus, it allows network designers to strike a balance between caching costs and overhead, on the one hand, and the risk of violating Consumer's delay, on the other hand, during prediction uncertainties.

We then derive a deterministic equivalent form for the *probabilistic* CCP model using Scenario Approximation (SA). In essence, the Probability Mass Function (PMF) of the predicted *interests* is used to replace the random variables, resulting in a closed form Integer Linear Programming (ILP) model that can be solved by numerical optimization. This enables deriving benchmark solutions for future *robust* predictive schemes and redefines the prediction gains under real-world uncertainties [13].

The remainder of the chapter is organized as follows. In Section 6.1, a background on stochastic optimization and CCP is given. The benchmark is introduced in Section 6.2. In Section 6.3, the stochastic formulation and deterministic equivalent are

proposed. A real-time low complexity guided heuristics is proposed in Section 6.4. The performance evaluation of the proposed framework is presented and discussed in Section 6.5 and 6.6. Finally, we conclude our findings in Section 6.7.

## 6.1 Background on Stochastic Optimization

*Robust* optimization refers to a class of optimization theory in which the formulated programming model contains uncertain variables [12,14,54]. In principle, two schemes are used namely *fuzzy* and *stochastic* optimization. The former models the uncertain information as fuzzy numbers with membership functions to capture the variations. The *stochastic*, on the contrary, models the uncertain information as random variables whose Probability Density Function (PDF) or PMF is used to represent the possible outcomes of prediction. Despite its simplicity, the *fuzzy* approach is more *conservative* in that it over-satisfies the constraints and degrades the value of the objective function [7, 22]. Hence, we focus on *stochastic* optimization which handles the uncertainty in constraints by CCP technique. In essence, CCP transforms the constraint with random variables into a *probabilistic* model that ensures its satisfaction by a minimum degree denoted by $\beta \in [0, 1]$.

In order to obtain a closed form solution, a deterministic equivalent is typically derived based on the statistical model of the random variable. The deterministic form can be obtained by means of Gaussian Approximation (GA), Scenario Approximation (SA) or *Markov Inequality*, among others [54, 68]. GA assumes that uncertain *data* is a normally distributed random variable, and thus uses the inverse of the Cumulative Density Function (CDF), variance and the satisfaction level $\beta$ to obtain a deterministic form. However, it results in a Second Order Cone Programming (SoCP)

model that is not necessarily convex over the range of $\beta$ [10]. Markov Inequality, on the other hand, results in a linear model representing the tail bound of the random variable. Yet more effort should be done to derive the empirical coefficients required for such an approximation. SA adopts the PMF of random variable to construct possible scenarios of the optimization model. The resultant model guarantees that the solution of such approximation satisfies the scenarios with a probability sum of $\beta$.

In this framework, we adopt the SA as it promises a solution that always satisfies the constraints by the predefined level $\beta$. This is in addition to its capability of obtaining an ILP model that does not change the order of magnitude of the problem at hand.

## 6.2 Delay-sensitive Formulation

The formulation provided in Section 5.2 aims to minimize the delay of *interests* and bounds the handover messages with a threshold. However, this does not guarantee that the delay is bounded, especially in the case of a network with limited resources. Therefore, in this formulation we minimize the total overhead and bound the delay.

Assuming the same system model explained in Section 4.2, finding the best possible placement of data with bounded delay can be formulated as follows:

$$\min_{\delta_r^d, \rho_r^m} \sum_{d \in D} \sum_{r \in R} \mathbb{P}_d^r \tag{6.1}$$

S.t.

$$5.2, 5.3, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10$$

$$\sum_{r \in R} |P_{u(d) \to r}| \delta_r^d < \gamma \quad \forall d \in D \tag{6.2}$$

The objective function (6.1) minimizes the summation of the total path updates required to cache all data $d$ in order to minimize the network overhead. Constraint (6.2) bounds the number of hops for every *data* $d$ by a threshold $\gamma$ which is an application specific delay requirement. Furthermore, all other constraints related to cache capacity from Chapter 5 hold here as well.

## 6.3 Robust Proactive Producer Mobility

### 6.3.1 Problem Definition and System Model

Both future mobility and *data* are used to predict the *interests* that will be dropped due to Producer handover. Given the set of *interests*, we find the optimal *data* placement in caches such that generated *interests* are cached and delivered with minimal overhead and bounded delay. Since predictors do not generate information with perfect accuracy, the estimated set of *interests* can be erroneous. Therefore, we use stochastic optimization to provide a robust optimal solution to the *data* placement

problem given the possible realizations of each *interest* from the predictor. The robust stochastic model for the formulation in Equations (6.1) and (6.2).

Assuming the system model in Section 4.2, each predicted *data* $d$ is represented as a random variable $\tilde{d}$ to model the prediction uncertainty. Each realization of this random variable, e.g., represents a video content encoded in different quality, is denoted by $s \in S_d$ and has probability $\pi^{d,s}$.

### 6.3.2 Stochastic Formulation

We formulate the *data* placement problem using CCP in which the objective is to minimize the overhead while the delay is bounded by a probabilistic level under limited cache capacity. The decision variables determine where to place the *data* and which *data* to remove from the caches. The first decision variable is denoted by $\delta_r^{\tilde{d}}$ which equals 1 if the erroneous predicted *data* $\tilde{d}$ will be stored in cache $r$, and equals 0 otherwise. The second decision variable is $\rho_r^m$, which decides on how many items should be removed from the cache to provide a space for the new *data*. The complete robust stochastic formulation is depicted as:

$$\min_{\delta_r^{\tilde{d}},\rho_r^m} \sum_{\tilde{d}\in D}\sum_{r\in R} \mathbb{P}_{\tilde{d}}^r \delta_r^{\tilde{d}} \tag{6.3}$$

S.t.

$$\sum_{r\in R} \delta_r^{\tilde{d}} \leq 1, \quad \forall \tilde{d} \in D \tag{6.4}$$

$$Pr\left\{\sum_{r\in R} |P_{u(\tilde{d})\to r}| \delta_r^{\tilde{d}} \leq \gamma\right\} \geq \ \beta, \ \forall \tilde{d} \in D \tag{6.5}$$

$$\sum_{m=0}^{c_r} \rho_r^m = 1, \quad \forall r \in R \tag{6.6}$$

$$c_r + \sum_{\tilde{d}\in D} \delta_r^{\tilde{d}} - \sum_{m=0}^{c_r} m\rho_r^m \leq c_r^{max}, \quad \forall r \in R \tag{6.7}$$

$$\sum_{\tilde{d}\in D} \delta_r^{\tilde{d}} - \sum_{m=0}^{c_r} m\rho_r^m \geq 0, \quad \forall r \in R \tag{6.8}$$

$$\sum_{m=0}^{c_r} m\rho_r^m < \alpha, \quad \forall r \in R \tag{6.9}$$

$$(\sum_{m=0}^{c_r} \rho_r^m k_r^m - \sum_{\tilde{d}\in D} \delta_r^{\tilde{d}}.\epsilon_{\tilde{d}})/K_r < \omega \quad \forall r \in R \tag{6.10}$$

$$c_r^{Sp} + \sum_{\tilde{d}\in D} \delta_r^{\tilde{d}} \leq c_r^{SpMax} \quad \forall r \in R \tag{6.11}$$

The objective function in Equation (6.3) minimizes the amount of overhead generated by the scheme. In particular, it minimizes the total summation of path updates messages to cache the required *data* in the network. Review Section 5.2 for more details on $\mathbb{P}_{\tilde{d}}^r$.

The deterministic constraints have been defined before in Section 5.2. The probabilistic constraint in (6.5) ensures that the network caches enough content for each

predicted *data* such that the delay requirement $\gamma$ is satisfied by a minimum level $\beta$. Indeed, the above formulation does not have a closed form solution due to the random variable and probabilistic constraint. Thus, we propose the deterministic equivalent model below.

### 6.3.3 Deterministic Equivalent

The stochastic formulation is replaced by the following deterministic equivalent using SA. In essence, the SA adopts the PMF of the *data* realizations to replace the random variable $\tilde{d}$. Herein, the first decision variable will thus become $\delta_r^{d,s}$ which equals 1 if the

optimizer decides to cache realization $s$ of *data* $d$ in router $r$, and equals 0 otherwise.

$$\min_{\delta_r^{d,s},\rho_r^m} \sum_{d\in D}\sum_{s\in S_d}\sum_{r\in R} \mathbb{P}_{d,s}^r \delta_r^{d,s} \tag{6.12}$$

S.t.

$$\sum_{r\in R} \delta_r^{d,s} \leq 1, \quad \forall d \in D, \forall s \in S \tag{6.13}$$

$$\sum_{r\in R} |P_{u(d,s)\to r}| \delta_r^{d,s} \leq \gamma \quad \forall d \in D, \forall s \in S_d \tag{6.14}$$

$$\sum_{s\in S_d}\sum_{r\in R} \delta_r^{d,s} \pi^{d,s} \geq \beta \quad \forall d \in D \tag{6.15}$$

$$c_r + \sum_{d\in D}\sum_{s\in S_d} \delta_r^{d,s} - \sum_{m=0}^{c_r} m\rho_r^m \leq c_r^{max}, \quad \forall r \in R \tag{6.16}$$

$$\sum_{d\in D}\sum_{s\in S_d} \delta_r^{d,s} - \sum_{m=0}^{c_r} m\rho_r^m \geq 0, \quad \forall r \in R \tag{6.17}$$

$$(\sum_{m=0}^{c_r} \rho_r^m k_r^m - \sum_{d\in D}\sum_{s\in S_d} \delta_r^{d,s}.\epsilon_{d,s})/K_r < \omega \quad \forall r \in R \tag{6.18}$$

$$c_r^{Sp} + \sum_{d\in D}\sum_{s\in S_d} \delta_r^{d,s} \leq c_r^{SpMax} \quad \forall r \in R \tag{6.19}$$

$$6.6, 6.9$$

The constraints in (6.14) and (6.15) are used to replace the probabilistic constraint (6.5). In essence, (6.14) selects the realizations to be cached for each *data* such that the delay constraint is satisfied. This is complemented by (6.15) which ensures that the total probability of cached realizations, for each *data*, surpasses the minimal level $\beta$. Moreover, the new objective function (6.12) and constraints in (6.16) and (6.17) are replacing the corresponding stochastic equations by using the *data* realizations

$(d, s)$ instead of the random variable $\tilde{d}$.

In the above formulation, the value of $\beta$ provides a trade-off between the risk of violating the delay requirement and the network gain quantified by the overhead in the objective function. A high value of $\beta$ forces the network to cache more *data* realizations in order to increase the likelihood of satisfying the delay requirement under erroneous prediction. However, such greedy caching results in high overhead which compromises the network gains. On the other hand, choosing a low value for $\beta$ results in caching fewer realizations to minimize the objective function. This strategy increases the risk of missing the actual future *data*, resulting in high chance of violating the delay constraint (6.14).

The objective function and all the constraints are now linear. Moreover, the values of the decision variables are either 0s or 1s. Hence, the optimization problem is a 0-1 Integer Linear Program (0-1 ILP) which can be solved by numerical optimization techniques and commercial solvers.

## 6.4 Robust Real-time Algorithm

In this section, we tackle the complexity of the optimal solution by proposing a robust real-time algorithm to find a near-optimal solution. The algorithm finds the optimal placement for each *interest* one at a time. By finding the local optimal of each *interest* a global near-optimal solution can be found. Using the same idea from Section 5.4 and Algorithms 1 and 3, the pseudo-code in Algorithm 4 is designed to obtain a solution for the robust formulation.

As discussed in Section 6.3.1, for each *data* $d$ a set of realizations $d_s$ are created and added to the list of *data* (now called $\tilde{D}$) to be optimized. This is shown in lines

3-5 in Algorithm 4. Since *data* placement is optimized one at time, the order of *data* impacts the final solution. Therefore, we sort the realizations of each *data* $d$ in the set $\tilde{D}$ descendingly based on the value of $\pi^{d,s}$. Specifically, realization $s_1$ of data $d$ comes before realization $s_2$ of the same data if $\pi^{d,s_1} \geq \pi^{d,s_2}$. This strategy guarantees that the interest realizations with high likelihood to be requested are placed first. As such, in high load scenarios (limited caching resources compared to interest realizations), the interests with high likelihood will be placed in the caching routers closer to the consumers. Thus, increases the possibility of satisfying the delay requirement set by the probabilistic constraint.

A summation variable $Sums$ is created for each *data* $d$ to keep track of the total summation of $\pi^{d_s}$ considered up to this point. Whenever $Sums[d]$ is greater than $\beta$, no more realizations needed to be optimized unless caching this realization does not add new path updates as overhead to the network. By doing so, the algorithm avoids the increase in overhead in order to optimize the objective function on the one hand, while trying to maximize the chance of satisfying the probabilistic constraint by adding more realizations that do not deteriorate the prediction gains.

For each *data* $d$ to be optimized, the set of routers is sorted based on the overhead of path updates if this router is used. Then, the first feasible router in the sorted list is chosen to cache this *data*. The complexity of the new algorithm is $O(|\tilde{D}|.|R|.log(|R|))$ where $|R|log|R|$ is the complexity of line 11 in which the routers are sorted. This process is repeated for all realizations denoted by $|\tilde{D}|$ calculated as $|D|\times$ the number of realizations.

**Algorithm 4** Pseudo-code of *RCacheMob-RT* scheme.

**Input:** *network*: The current state of the network
**Input:** $D$: Set of requests to be optimized

```
1: Initialize set D̃ ← φ
2: Create Sums[d] = 0 ∀d ∈ D
3: for all d ∈ D do
4:     Add realizations d_s of data d to D̃
5: end for
6: Sort D̃ such that d_{s_1} comes before d_{s_2} if π^{d_{s_1}} ≥ π^{d_{s_2}}
7: Initialize Sol.DataAdded[r] = 0 ∀r ∈ R
8: Initialize Sol.DataRemoved[r] = 0 ∀r ∈ R
9: GetNetworkMetrics(network,D,R)
10: for all d_s ∈ D̃ do
11:     R' ← SortRouters(d_s, R)
12:     for all r ∈ R' do
13:         if Sums[d] ≥ β and pathUpdates[d_s][r] > 0 then
14:             Break
15:         end if
16:         Sol' ← Sol
17:         Sol'.DataAdded[r] ← Sol'.DataAdded[r] + 1
18:         if C_r + Sol.DataAdded[r] − Sol.DataRemoved[r] = C_r^max then
19:             Sol'.DataRemoved[r] ← Sol'.DataRemoved[r] + 1
20:         end if
21:         if Sol'.isFeasible(d_s,r) then
22:             Sol ← Sol'
23:             Sol.SelectedRouter[d_s] ← r
24:             Break
25:         end if
26:     end for
27:     Sums[d] ← Sums[d] + π^{d_s}
28: end for
29: ApplySolution(Sol)
```

## 6.5 Performance Evaluation of Non-robust Optimal Scheme

In this section, we evaluate *OpCacheMob* benchmark under perfect prediction to measure the maximum prediction gain.

### 6.5.1 Experimental Setup

The experimental setup used is explained in Section 3.4. For every experiment one factor is varied to test the impact of it on the performance of the scheme. *OpCacheMob* is controlled by the optimization window (set to 2 seconds) and four thresholds to bound the overhead. After tuning the parameters to guarantee that a feasible caching solution can be attained under the current network configuration, the thresholds are set to the following:

1. $\gamma$: controls how far the *data* will be placed from the Consumer side. Similar to the previous threshold, it is based on the average number of hops to reach the Producer. Thus, the resulted delay will not be longer than the one to reach the Producer.

2. $\alpha$: the maximum number of replacements per router is set to 10% of the maximum cache size.

3. $\omega$: the total content value cached in a router should not decrease by more than 10% of its initial total content value before applying our mobility-driven caching/replacement strategy.

### 6.5.2 Comparative Schemes

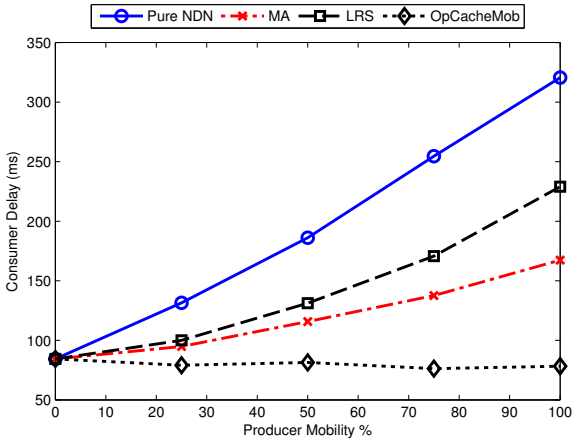
We evaluate *OpCacheMob* and compare it with three mobility management approaches. Namely, NDN with no scheme (Pure-NDN), Mobility Anchor (*MA*) represented by [62] and Location Resolution Scheme (*LRS*) represented by [57].

### 6.5.3 Producer Mobility

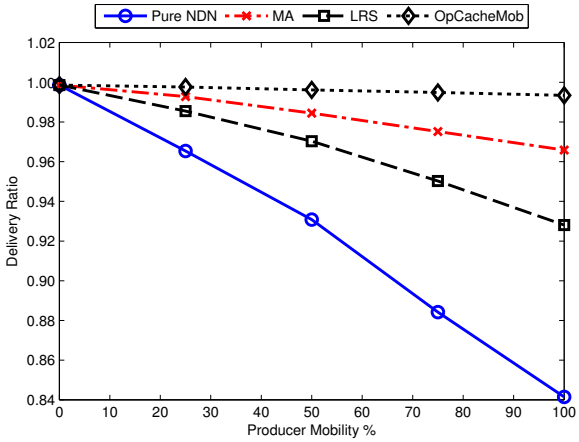
The performance of *OpCacheMob* while changing the percentage of mobile Producers is shown in Figures 6.1a, 6.1b and 6.1c. In particular, the average delay is stable irrespective of the mobility percentage which shows the optimality of the proposed benchmark. Likewise, the delivery ratio is almost 100% because the scheme avoids retransmissions by proactively caching the *data* prior to handover. As a consequence, the overhead is compromised to provide the stability which led to higher number of control messages compared to the non-predictive schemes *MA* and *LRS*. Discussion about the performance of those schemes was provided in Section 5.5.

### 6.5.4 Cache Size

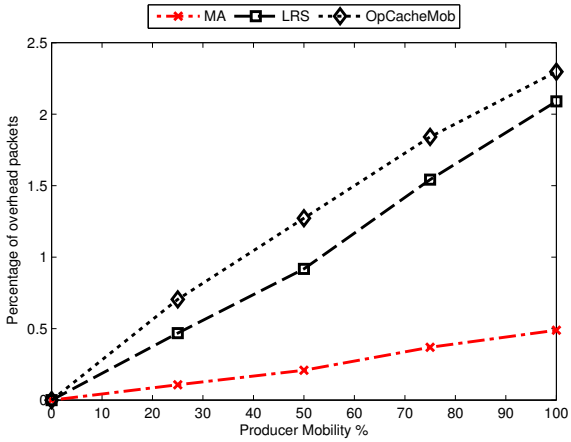
Analyzing *OpCacheMob* under different cache resources resulted in a reduced average Consumer delay as shown in Figure 6.2 when 50% of Producers are mobile. As depicted, the *OpCacheMob* behaves as the *pure-NDN* when there is no space available in the cache (i.e., cache percentage = 0). However, *OpCacheMob* efficiently utilizes any minor availability in the cache which significantly improves the Consumer's delay as shown in Figure 6.2. Such degree of enhancement increases with the availability of more caching space where more data for the mobile Producers can be stored close to their requesting user. As such, the *OpCacheMob* continues to provide a lower bound




(a) Showing the average Consumer delay.




(b) Showing the delivery ratio.




(c) Showing the overhead percentages.

Figure 6.1: *OpCacheMob* compared to other schemes under different mobility percentages

for delay of the existing non-predictive *MA* and *LRS*.

### 6.5.5 Sensitivity Analysis

As shown in the previous chapter, the formulation is sensitive to error in requests predictions. Robust schemes solve this problem as will be shown in the next section.

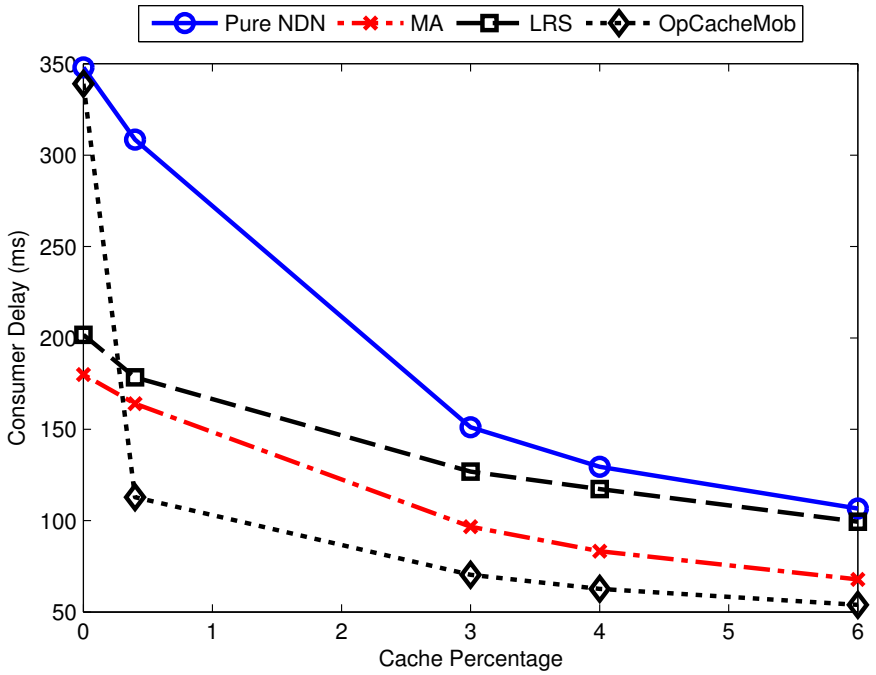



Figure 6.2: *OpCacheMob* compared to other schemes under different cache percentages, showing the average consumer delay at 50% Producer mobility.

## 6.6 Performance Evaluation of Robust Proposals

### 6.6.1 Introducing Prediction Errors in the Assessment Framework

Prediction uncertainty is simulated in the experiment design by adding errors to *interests* names. In particular, a set of *interests* is randomly chosen to be erroneous. The size of the set is based on the factor *percentage of error*, where 0% error percentage means no *interest* is erroneous and 100% means all *interests* are wrongly predicted. Then, the names of these *interests* are changed, according to the Zipf distribution, to a new random, yet valid, name.

### 6.6.2 Comparative Schemes

We evaluate the proposed *stochastic* schemes and compare it with: 1) the *non-proactive* Mobility Anchor (*MA*) scheme [62], and 2) the *non-robust* proactive benchmark proposed in Section 6.2 which caches the most probable *data* realization.

### 6.6.3 Overhead Aggregation

The process of applying the solutions to the network was discussed in Section 5.4.2. In this section, we explain how the overhead can be reduced by aggregating the update messages. For every optimization window, the control messages will be sent from the MSU to routers. Instead of sending a separate control *interests* for each required action, the actions meant to be executed on the same router are aggregated in one control message. Hence, each router (if it is involved in the caching decision) will receive one control message with all the actions to be taken to proactively support mobility.

### 6.6.4 Robustness Gains and Costs

We discuss the impact of errors on the schemes, where the probabilistic level $\beta$ is set to a relatively high value of 0.95 that quantifies the performance bounds of adopting a *robust* scheme. Figure 6.3 depicts the average Consumers delay for different error percentages. In the idealistic scenario, without prediction errors, *Op-CacheMob* (*non-robust*), *RCacheMob-Op* (*robust*) benchmarks and the heuristic algorithm, *RCacheMob-RT*, have a lower delay than the *MA* scheme. Specifically, the delay is reduced by 65% for *OpCacheMob* compared to the *MA* scheme. This gap in performance shows the benefits of proactive caching in the network prior to mobility

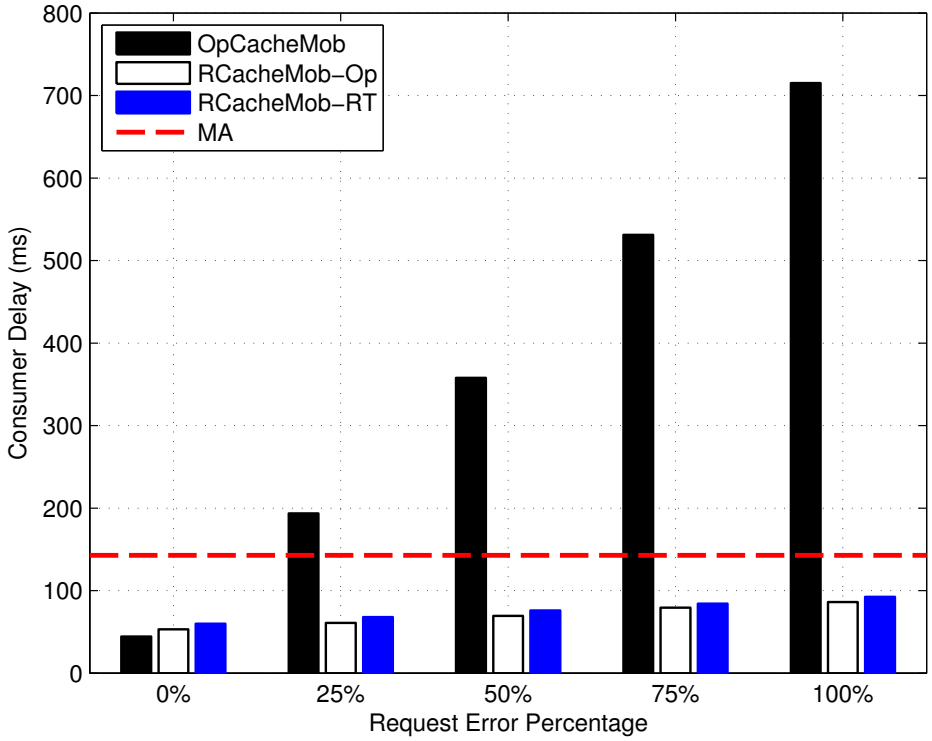



Figure 6.3: The average Consumers delay for *robust* and *non-robust* schemes with different error percentages. The *MA* approach has a fixed delay of 142ms.

events.

When errors are introduced to the *data* prediction, the *OpCacheMob* scheme fails to maintain the performance gains. Hence, the delay is increased by 30% and 377% at 25% and 100% error percentages, respectively, compared to the error-free scenario. Assuming idealistic scenarios, by *OpCacheMob* scheme, resulted in caching only the most probable *data* while ignoring other realizations. The actual requested *data*, which are not cached, will be sent after the Producer moves to the new location and sends path updates to the network. Hence, it caused more delay than the *non-proactive MA* scheme which forwards the *interest*, after handover, to the Producer in

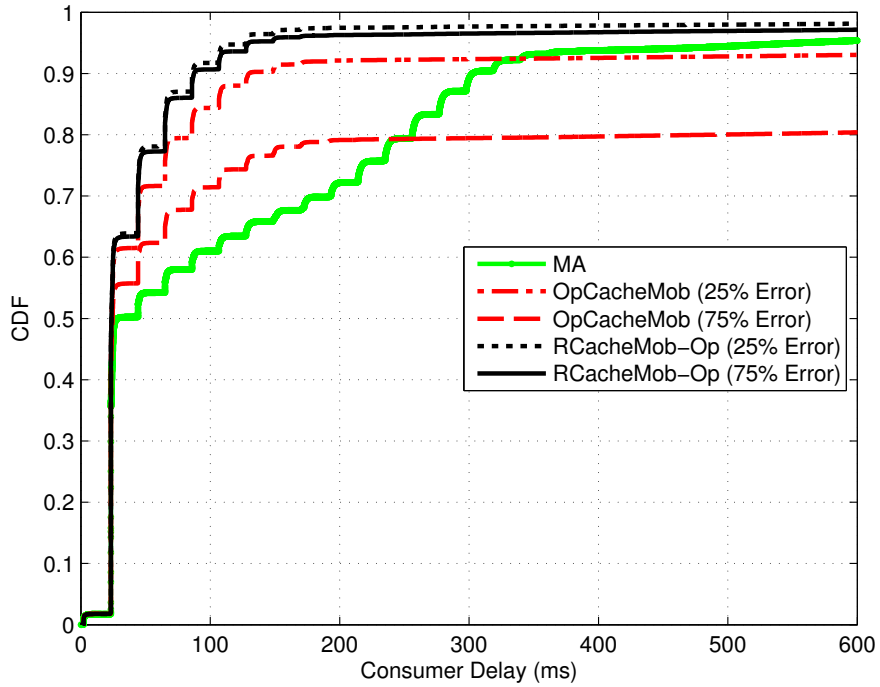



Figure 6.4: The CDF of *MA*, non-robust and *robust* schemes with 25% and 75% error percentages.

the new network. On the contrary, the proposed *robust* scheme cached more realizations to maintain the Consumers delay. Hence, it achieves a delay based *prediction gain* of 28% in the worst case scenario (i.e., 100% prediction error) when compared to the *MA* scheme.

The CDFs of the Consumers delay using the mobility anchor, non-robust and robust benchmarks are depicted in Figure 6.4 with 25% and 75% prediction request error. While the *non-robust* outperformed the *MA* in the case of low prediction error (i.e., 25%), a higher error of 75% resulted in a non-tolerable maximum delay (more than $600ms$) by the former in 20% of the cases. Unlike *OpCacheMob* and *MA* schemes, the *RCacheMob-Op* achieved a stable performance with a $90^{th}$ percentile of Consumers delay below $100ms$. This is in addition to capping the maximum gain

Table 6.1: The overhead percentages at 50% mobility and 50% error in requests prediction for all schemes.

| | | **Overhead percentage** | |
|---|---|---|---|
| | | Without aggregation | With aggregation |
| *OpCacheMob* | | 3.25% | 1.74% |
| *RCacheMob-Op* | $\beta = 0.95$ | 50.64% | 2.09% |
| | $\beta = 0.85$ | 33.54% | 2.01% |
| | $\beta = 0.75$ | 26.98% | 1.98% |
| *RCacheMob-RT* | $\beta = 0.95$ | 59.20% | 1.26% |
| | $\beta = 0.85$ | 40.44% | 1.25% |
| | $\beta = 0.75$ | 30.04% | 1.18% |
| Mobility Anchor | | 2.38% | |

below the *MA* scheme in all cases.

The above gains of robustness come at the cost of increased overhead due to the control packets needed for proactive caching of *data*. The overhead percentages of the four schemes are summarized in Table 6.1 for different satisfaction levels. Moreover, the regular overhead (i.e., number of path updates) and the aggregated control messages are reported as well. In the case of *RCacheMob-Op*, the overhead is 2.09% which is lower than the *MA* scheme. In essence, *MA* sends many control packets for every handover event, while the *robust* scheme sends aggregated control packets containing the routing information which is broadcast in the network every optimization window, see Section 6.6.3. However, the *OpCacheMob* overhead is less than *RCacheMob-Op* by 17%. These extra control packets sent by *RCacheMob-Op* scheme, referred to as the cost of robustness, are due to caching more *data* to handle the uncertainties of prediction.

The performance of *RCacheMob-RT* is close to that of *RCacheMob-Op* benchmark

with a slight increase in Consumer delay. In particular, the increase in delay using the heuristic algorithms ranges from 7.4% to 13%. This near-optimal performance comes with a cost of extra path updates messages (i.e., overhead) as shown in Table 6.1. The extra 18% of overhead messages can be reduced when aggregation is used, see Section 6.6.3. Moreover, this can be compensated by the reduction in complexity of *RCacheMob-Op*. On the other hand, the aggregated overhead of *RCacheMob-RT* is less than *RCacheMob-Op*. This is because the heuristic algorithm tends to use fewer routers to proactively cache the *data* than the optimizer. In essence, the former fully exploits the router with the least overhead and only chooses the next one when no space is available, whereas the optimizer chooses multiple routers as along as the overhead is minimal.

However, the increase of overhead saves 83% of *interests* retransmissions, where the retransmission ratios in 100% error are 0.85 and 0.15 for *OpCacheMob* and *RCacheMob-Op* schemes respectively. This metric is an indication of how effective the *RCacheMob-Op* scheme is in avoiding *interests* dropping to guarantee optimal utilization of network resources. Moreover, the ratio in the *non-robust* scheme is enormously increased from 0.01 to 0.85 because of failing to cache the accurate *data*. On the other hand, the *robust* scheme has 57% fewer retransmissions than the *non-proactive MA* scheme. This is due to the reactive approach taken by *MA* which leads to retransmissions when the Producer is in the registration phase.

### 6.6.5 Probabilistic Level Design Trade-off

This experiment is designed to evaluate the effect of the probabilistic threshold $\beta$ on both the costs and gains of the *robust* schemes. In particular, the schemes are tested

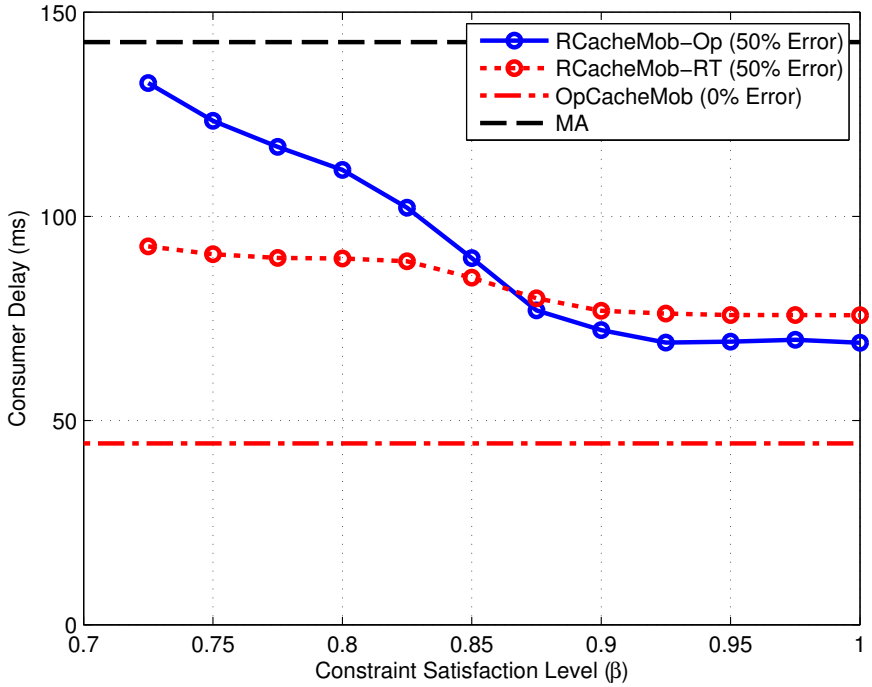



Figure 6.5: The average Consumers delay of the *robust* scheme at 50% error percentage with different values of the threshold $\beta$.

under 50% of prediction error and using $\beta$ ranging from 0.725 to 1. As depicted in Figure 6.5, the delay experienced by the Consumers (i.e., robustness gain) gets shorter when $\beta$ is increased. Moreover, there is no noticeable gain after $\beta$=0.875, since the number of realizations chosen is sufficient to satisfy Constraint (6.15). The *MA* scheme has a delay of $142ms$ which is an upper delay bound to the *robust* scheme for a fixed prediction error and mobility percentages. On the other hand, the idealistic scenario is simulated by using the *non-robust* scheme with no prediction errors, thus provides the lower bound for the delay (i.e., maximum prediction gains) which is found to be $46ms$. With regards to *RCacheMob-RT*, the robustness gain is approximately constant. This is because the heuristic algorithm is slightly conservative with respect of the delay. In particular, it allows caching more data to guarantee meeting the

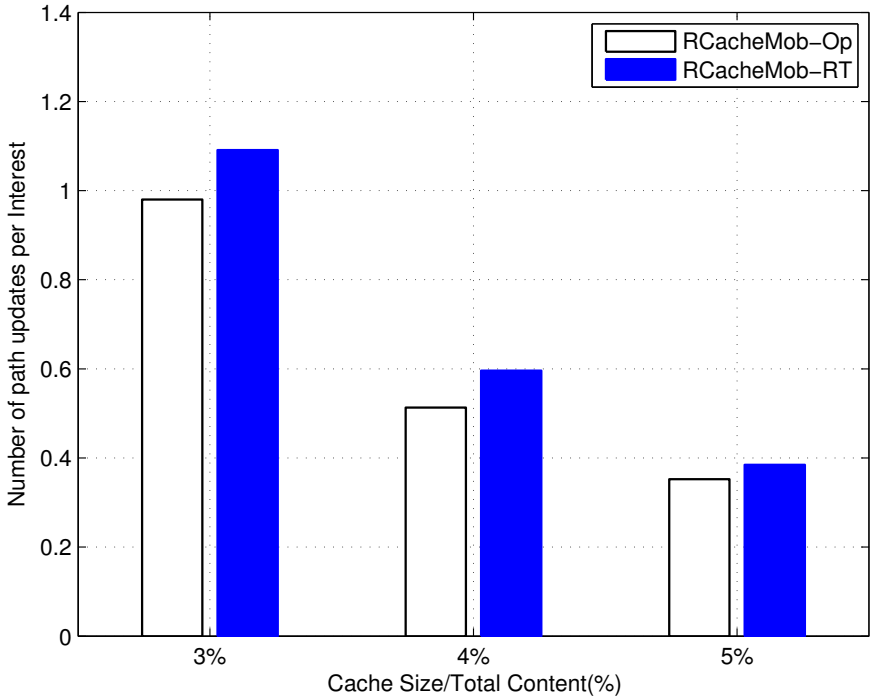



Figure 6.6: The number of path updates per *interest* for proactive schemes with three different cache sizes.

application requirements.

With respect to the overhead, 3 different values for $\beta$ are illustrated in Table 6.1. As shown, the overhead increases with the selected value of $\beta$ due to caching more *data* to satisfy the delay constraints 6.14 and 6.15. However, such increase is negligible and maintains the superiority of the proactive *robust* scheme over the non-proactive *MA*. In essence, the value of probabilistic level $\beta$ allows network designers to control the trade-off between robustness gains, represented by the Consumers delay in our case, and the robustness costs represented by network overhead.

### 6.6.6 *RCacheMob-RT* vs. *RCacheMob-Op*

Here, we compare the performance of the guided heuristics scheme, *RCacheMob-RT*, to the robust optimal, *RCacheMob-Op*, using different cache sizes. Figure 6.6 depicts the number of path updates per *interest* required by each scheme with three cache sizes (3%, 4% and 5%). Moreover, this type of overhead is the value of the objective function Equation (6.12). As shown, *RCacheMob-RT* is giving a near-optimal performance where the optimality gap is ranging between 9% and 16%. In fact, the gap decreases when more resources (i.e., bigger cache) is used since the algorithm of *RCacheMob-RT* has more space to find a near-optimal solution.

The Consumer delay is shown in Figure 6.7 where the gain of prediction is maintained in all cases such that it does not violate the application delay requirement. In particular, the difference in delay between the two schemes is between 2.9% in the case of low resources and 12.5% in high resources.

## 6.7 Conclusions

This chapter proposed a robust optimal Producer mobility management scheme that exploits in-network resources and data prediction. Under different degree of prediction uncertainties, the results demonstrated the ability of the proposed scheme to proactively cache future *data* and avoid *interests* dropping. By generating 2% of control packets, the scheme is successful at maintaining lower delays compared to the *non-robust* scheme primarily proposed in the literature under idealistic prediction. Moreover, the number of *interests* retransmissions is minimal since the *interests* are satisfied before any handover. Nevertheless, the performance gap between the existing mobility management solutions and the *proactive* schemes can be controlled by

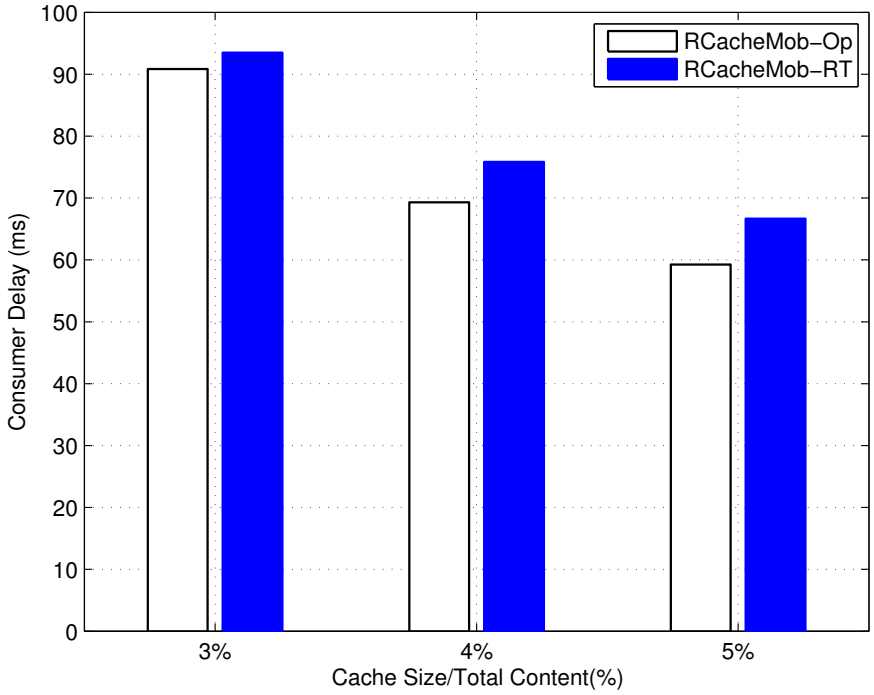



Figure 6.7: The average Consumer delay for proactive schemes under three different cache sizes.

the probabilistic level defined in our proposed *RCacheMob-Op*. Such outcomes also provide new trade-off between the cost of increasing the overhead and the risk of violating the Consumers delay in the light of the prediction uncertainty.

# Chapter 7

# Conclusions

## 7.1 Thesis Summary

In this thesis, we demonstrate the impact of mobility on Named Data Networking (NDN) and how location and content predictions can be used to provide seamless mobility with reliable and quality of service-aware content delivery. Instead of reactively solving the mobility problem, the use of predictions can aid in taking proactive actions. We presented how the user's Quality of Service (QoS) is stable during both Consumer and Producer mobility by using the proposed proactive schemes.

We provide both qualitative and numerical evaluation of state-of-the-art mobility support schemes in ICN. A comprehensive assessment framework is used to reveal the limitations of reactive schemes and quantify the preliminary gains of a proactive approach under idealistic scenarios. Moreover, the framework is used for sensitivity analysis that unleashes the impact of prediction errors on the performance of proactive schemes.

Consumer mobility is tackled in Chapter 4 where the problem is defined, formulated and an optimal solution is obtained. Mainly, we propose an optimal proactive

mobility support scheme, *OpCCMob*, that uses predictions to cache data near the Consumer prior to the handover event. The performance of *OpCCMob* compared to the two main approaches of Consumer mobility (Reactive and Semi-proactive) provides a gap that can be filled by utilizing more cache resources. On the other hand, the overhead associated with the scheme is considered as an upper-bound to other schemes.

In Chapter 5, optimal and real-time solutions are proposed to support Producer mobility. In the first part, a formulation to the problem is presented which minimizes the delay experienced by the Consumer and limit the handover. The optimal solution to this formulation can be used as a benchmark, called *OpProMob*, which is successful to provide constant delay during handover events. Nevertheless, the performance gap between the existing mobility management solutions and the *OpProMob* can be utilized to select the optimal cache size that guarantees the maximum tolerated delay. The second part of Chapter 5 demonstrates a practical real-time solution to the same problem. Using a greedy approach, the algorithm is successful to provide a near-optimal with lower complexity.

In the first part of Chapter 6, we propose a delay-sensitive alternative to the formulation in Chapter 5. The goal is to find the optimal placement of content without violating application delay requirements. In the second part, a robust framework is proposed to handle errors in predictions. First, a stochastic optimization is used to find an optimal placement under erroneous predicted information. The results show that the stochastic scheme achieves a delay based gain of 28% compared to literature in the worst scenario where all Producers are mobile. Then, a heuristic approach is proposed to find a near-optimal solution with reduced complexity. By compromising

the overhead on the network by 10%, the scheme can maintain the predication gain of the optimal scheme and perform in real-time. Moreover, the obtained solution is deemed robust as it satisfies the delay requirement at a probability more than the defined QoS level $\beta$. Nevertheless, the overhead shows a small optimality gap of 12% compared to the high complexity optimal solution obtained by commercial solvers. Such gap maintains the superiority of proactive schemes over state-of-the-art reactive schemes under prediction uncertainties.

## 7.2 Future Directions

The following are some future directions that can be followed to support the implementation of proactive caching for mobility support in practice:

1. *Joint Consumer & Producer mobility*:

   The research proposed in this thesis tackles Consumer and Producer mobilities as two separate problems in order to evaluate their individual impact on network performance and assess the ability of proactive caching to handle each type. However, Consumers and Producers move in the network with no restrictions. Hence, a scheme that considers both types at the same time is worth investigating.

2. *Consider video applications*:

   As mentioned in Chapter 1, video content has a big share of the global Internet traffic. Moreover, streaming videos is considered as a delay-sensitive application and mobility of any of the Consumer or Producer may affect the overall experience of the user. Studying the network with such application can demonstrate

the real gain of the proposed schemes [11].

3. *Implement a proof-of-concept experiment*:

   One of the motivations behind choosing NDN architecture is the availability of a testbed and open-source software that uses NDN over the Internet. Our real-time scheme proposed in Chapter 6 can be implemented in one of the nodes that can proactively place data in the network.

4. *Consider other Stochastic Optimization methods*:

   The method used to obtain a deterministic form equivalent of the stochastic problem in Chapter 6 is Scenario Approximation. Other methods can be used that may provide different results which are worth looking at. For example, Gaussian Approximation (GA) that uses the inverse of Cumulative Density Function (CDF), variance and satisfaction level $\beta$ to find a deterministic form. Such comparison provides guidelines to service providers on how to choose the prediction error model and how much effort can be invested in improving the prediction [15].

5. *Consider Mobility Predictions errors*:

   In this research, we have proposed a robust scheme that considers erroneous request predictions, since errors in mobility predictions have less impact. Designing a fully resilient scheme that is robust to both types of predictions is an important next step, especially in scenario in which location prediction is challenging such as indoor network.

6. *Adaptive Robust Scheme*:

The level of satisfaction $\beta$ in the robust formulation provides a trade-off between Consumer experience and network overhead. In this work, we assume that $\beta$ is static and does not change. Considering an adaptive $\beta$ where the value changes over the optimization window based on a feedback from the network enables the service provider to control the trade-off and achieve a uniform user experience among the users.

## 7.3 Concluding Remarks

We conclude this thesis with the following remarks:

1. Supporting seamless mobility is a core property in any design of the future Internet.

2. Reactive mobility schemes can provide basic mobility support in NDN. However, the long delays and *interests* retransmissions generated in the process make it hard to support seamless user experience. This work has shown that there is a gap of performance between these schemes and the benchmark and this gap can be filled by utilizing the cache resources in the network.

3. Proactive proposals are promising schemes which can guarantee the QoS of users. However, they require extra consideration to handle errors in predictions. Stochastic optimization can provide a robust solution that fulfills the QoS requirements at the expense of a slight increase in network overhead (referred to as cost of robustness). Prediction gains are still retained compared to the state-of the art reactive solutions.

4. Commercial solvers can be used only to obtain the optimal placement for proactive caching, which can provide a benchmark for the optimal mobility scheme performance. However, polynomial-time solutions are essential for practical schemes since decisions should be taken frequently and in sub-seconds.

# Bibliography

[1] S. Abdelhamid, H. Hassanein, G. Takahara, and H. Farahat. Caching-assisted access for vehicular resources. In *IEEE Conference on Local Computer Networks (LCN)*, pages 28–36, September 2014.

[2] H. Abou-zeid, H. S. Hassanein, and R. Atawia. Towards mobility-aware predictive radio access: modeling; simulation; and evaluation in lte networks. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems(MSWIM)*, pages 109–116, 2014.

[3] H. Abu-Ghazaleh and A. Alfa. Application of mobility prediction in wireless networks using markov renewal theory. *IEEE Transactions on Vehicular Technology*, 59(2):788–802, February 2010.

[4] A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM: NDN simulator for ns–3. Technical Report NDN-0005, NDN, October 2012.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, July 2012.

[6] R. Atawia. *Robust Predictive Resource Allocation for Video Delivery Over Future Wireless Networks*. PhD thesis, Queen's University, August 2017.

[7] R. Atawia, H. Abou-zeid, H. Hassanein, and A. Noureldin. Robust resource allocation for predictive video streaming under channel uncertainty. In *IEEE Global Communications Conference (GLOBECOM)*, pages 4683–4688, December 2014.

[8] R. Atawia, H. Abou-zeid, H. Hassanein, and A. Noureldin. Chance-constrained QoS satisfaction for predictive video streaming. In *IEEE Conference on Local Computer Networks (LCN)*, pages 253–260, October 2015.

[9] R. Atawia, H. Abou-zeid, H. S. Hassanein, and A. Noureldin. Joint chance-constrained predictive resource allocation for energy-efficient video streaming. *IEEE Journal on Selected Areas in Communications*, 34(5):1389–1404, May 2016.

[10] R. Atawia, H. Hassanein, and A. Noureldin. Energy-efficient predictive video streaming under demand uncertainties. In *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[11] R. Atawia, H. Hassanein, and A. Noureldin. Optimal and robust qos-aware predictive adaptive video streaming for future wireless networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2017.

[12] R. Atawia, H. S. Hassanein, H. Abou-Zeid, and A. Noureldin. Robust content delivery and uncertainty tracking in predictive wireless networks. *IEEE Transactions on Wireless Communications*, 16(4):2327–2339, 2017.

[13] R. Atawia, H. S. Hassanein, N. A. Ali, and A. Noureldin. Utilization of stochastic modelling for green predictive video delivery under network uncertainties. *IEEE Transactions on Green Communications and Networking*, pages 1–14, 2017.

[14] R. Atawia, H. S. Hassanein, and A. Noureldin. Fair robust predictive resource allocation for video streaming under rate uncertainties. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, December 2016.

[15] R. Atawia, H. S. Hassanein, and A. Noureldin. Robust long-term predictive adaptive video streaming under wireless network uncertainties. *IEEE Transactions on Wireless Communications*, pages 1–14, 2017.

[16] A. Azgin, R. Ravindran, and G. Wang. Mobility study for named data networking in wireless access networks. In *IEEE International Conference on Communications (ICC)*, pages 3252–3257, June 2014.

[17] A. Azgin, R. Ravindran, and G. Wang. A scalable mobility-centric architecture for named data networking. *CoRR*, abs/1406.7049, June 2014.

[18] G. M. Brito, P. B. Velloso, and I. M. Moraes. *Information Centric Networks: A New Paradigm for the Internet*. Wiley, 2013.

[19] E. C. Perkins. IP mobility support for IPv4, revised. RFC 5944, 2010.

[20] CAC. Centre for advanced computing (CAC). http://cac.queensu.ca/, 2017. Accessed on September 29, 2017.

[21] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks (extended version). *Computer Communications*, 36(7):758–770, April 2013.

[22] Q. Chen. *Comparing probabilistic and fuzzy set approaches for designing in the presence of uncertainty*. PhD thesis, Virginia Tech, 2000.

[23] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *IEEE Computer*, 36(3):63–70, March 2003.

[24] D. Cheriton and M. Gritter. TRIAD: A new next-generation internet architecture, 2000.

[25] Cisco. Cisco visual networking index: Forecast and methodology, 2016–2021. White Paper, Cisco, June 2017.

[26] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021. White Paper, Cisco, February 2017.

[27] Cisco. The zettabyte era: Trends and analysis. White Paper, Cisco, June 2017.

[28] J. G. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, April 1984.

[29] A. Dabirmoghaddam, M. M. Barijough, and J. Garcia-Luna-Aceves. Understanding optimal caching and opportunistic caching at "the edge" of information-centric networks. In *Proceedings of the International Conference on Information-centric Networking*, pages 47–56, September 2014.

[30] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl. Network of information (NetInf) – an information-centric networking architecture. *Computer Communications*, 36(7):721–735, April 2013.

[31] N. Eagle and A. S. Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.org/mit/reality/, July 2005.

[32] M. Elazab, A. Noureldin, and H. S. Hassanein. Integrated cooperative localization for vehicular networks with partial gps access in urban canyons. *Vehicular Communications*, 9(Supplement C):242 – 253, 2017.

[33] H. Farahat, R. Atawia, and H. S. Hassanein. Robust predictive caching for producer mobility management in named data networking. Journal In preparation, 2017.

[34] H. Farahat, R. Atawia, and H. S. Hassanein. Robust proactive mobility management in named data networking under erroneous content prediction. [To be published in GlobeCom 2017], December 2017.

[35] H. Farahat and H. S. Hassanein. On the design and evaluation of producer mobility management schemes in named data networks. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems(MSWIM)*, pages 171–178, November 2015.

[36] H. Farahat and H. S. Hassanein. Optimal caching for producer mobility support in named data networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[37] H. Farahat and H. S. Hassanein. Supporting consumer mobility using proactive caching in named data networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2016.

[38] H. Farahat and H. S. Hassanein. Decentralized proactive caching for producer mobility management in named data networks. Journal In preparation, 2017.

[39] H. Farahat and H. S. Hassanein. Proactive caching for producer mobility management in named data networks. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 171–176, June 2017.

[40] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: incrementally deployable ICN. In *Proceedings of the ACM SIGCOMM*, pages 147–158, August 2013.

[41] FP7. Scalable and adaptive internet solutions (SAIL). http://www.sail-project.eu/, 2017. Accessed on September 29, 2017.

[42] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 310–315, March 2012.

[43] G. Garcia, A. Beben, F. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, and E. Hadjioannou. COMET: Content mediator architecture for content-aware networks. In *Future Network Mobile Summit*, pages 1–8, June 2011.

[44] J. Ge, S. Wang, Y. Wu, H. Tang, and Y. E. Performance improvement for source mobility in named data networking based on global–local fib updates. *Peer-to-Peer Networking and Applications*, 9(4):670–680, July 2016.

[45] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the ACM Workshop on Hot Topics in Networks*, pages 1:1–1:6, November 2011.

[46] P. Graham. Web 2.0. http://www.paulgraham.com/web20.html, 2005. Accessed on September 29, 2017.

[47] Gurobi. Gurobi optimizer reference manual. http://www.gurobi.com, 2015. Accessed on September 29, 2017.

[48] D. Han, M. Lee, K. Cho, T. Kwon, and Y. Choi. Publisher mobility support in content centric networks. In *International Conference on Information Networking (ICOIN)*, pages 214–219, February 2014.

[49] F. Hermans, E. Ngai, and P. Gunningberg. Global source mobility in the content-centric networking architecture. In *Proceedings of the ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, pages 13–18, June 2012.

[50] J. Ioannidis, D. Duchamp, and G. Q. M. Jr. IP-based protocols for mobile internetworking. In *SIGCOMM Proceedings of the Conference on Communications architecture and protocols*, pages 235–245, September 1991.

[51] A. Ioannou and S. Weber. A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Communications Surveys and Tutorials*, 18(4):28470–2886, Fourthquarter 2016.

[52] ITU. ICT facts and figures 2016. http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx, 2016. Accessed on September 29, 2017.

[53] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. *Communications of the ACM*, 55(1):117–124, January 2012.

[54] P. Kali and S. W. Wallace. *Stochastic Programming.* Springer, New York, NY, USA, 1994.

[55] K. Katsaros, G. Xylomenos, and G. C. Polyzos. Multicache: An overlay architecture for information-centric networking. *Computer Networks*, 55(4):936–947, March 2011.

[56] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, and I. Yeom. Mobility support in content centric networks. In *Proceedings of the ICN workshop on Information-centric networking*, pages 13–18, August 2012.

[57] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, and I. Yeom. End-to-end mobility support in content centric networks. *International Journal Communication Systems*, 28(6):1151–1167, April 2015.

[58] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Computer Communication Review*, 37(4):181–192, August 2007.

[59] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal of Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[60] K. Laasonen. Route prediction from cellular data. In *Proceedings of the second workshop on Context Awareness for Proactive Systems*, pages 147–158, June 2005.

[61] D. Lagutin, K. Visala, and S. Tarkoma. *Publish/Subscribe for Internet: PSIRP Perspective*, pages 75–84. Towards the Future Internet (Valencia FIA book), 2010.

[62] J. Lee, S. Cho, and D. Kim. Device mobility management in content-centric networking. *IEEE Communications Magazine*, 50(12):28–34, December 2012.

[63] J. Lee, D. Kim, M. Jang, and B.-J. Lee. Mobility management for mobile consumer devices in content centric networking (CCN). In *IEEE International Conference on Consumer Electronics (ICCE)*, pages 502–503, January 2012.

[64] M. B. Lehmann, M. P. Barcellos, and A. Mauthe. Providing producer mobility support in ndn through proactive data replication. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS2016)*, pages 383–391, April 2016.

[65] D. Li and M. C. Cuah. SCOM: A scalable content centric network architecture with mobility support. In *IEEE International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 25–32, December 2013.

[66] W. Li, S. M. A. Oteafy, and H. S. Hassanein. On the performance of adaptive video caching over information-centric networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[67] W. Li, S. M. A. Oteafy, and H. S. Hassanein. Rate-selective caching for adaptive streaming over information-centric networks. *IEEE Transactions on Computers*, 66(9):1613–1628, September 2017.

[68] W.-L. Li, Y. Zhang, A.-C. So, and M. Z. Win. Slow adaptive ofdma systems through chance constrained programming. *IEEE Transactions on Signal Processing*, 58(7):3858–3869, July 2010.

[69] Z. Liu, Y. Wu, E. Yuepeng, J. Ge, and T. Li. Experimental evaluation of consumer mobility on named data networking. In *International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 472–476, July 2014.

[70] Y. Luo, J. Eymann, and A. Timm-Giel. Mobility support for content centric networking. *Telecommunication Systems*, pages 1–18, June 2014.

[71] K. Mahdi, H. Farahat, and M. Safar. Temporal evolution of social networks in paltalk™. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '08, pages 98–103, 2008. ACM.

[72] K. Mahdi, M. Safar, and H. Farahat. Analysis of temporal evolution of social networks. *Journal of Mobile Multimedia*, 5(4):333–350, December 2009.

[73] NDN. Named data networking testbed. https://named-data.net/ndn-testbed/, 2017. Accessed on September 29, 2017.

[74] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. M. Correia. The way 4WARD to the creation of a future internet. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, September 2008.

[75] ns–3. Network simulator 3. http://www.nsnam.org/, 2013. Accessed on September 29, 2017.

[76] A. Passarella. A survey on content-centric technologies for the current internet: CDN and P2P solutions. *Computer Communications*, 35(1):1–32, January 2012.

[77] H. Pinto, J. M. Almeida, and M. A. Gonçalves. Using early view patterns to predict the popularity of youtube videos. In *ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 365–374, February 2013.

[78] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the ICN workshop on Information-centric networking*, ICN '12, pages 55–60, August 2012.

[79] I. Psaras, W. K. Chai, and G. Pavlou. In-network cache management and resource allocation for information-centric networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):2920–2931, November 2014.

[80] R. Ramjee, K. Varadhan, L. Salgarelli, S. Thuel, S.-Y. Wang, and T. La Porta. HAWAII: a domain-based approach for supporting mobility in wide-area wireless networks. *IEEE/ACM Transactions on Networking*, 10(3):396–410, June 2002.

[81] Y. Rao, H. Luo, D. Gao, H. Zhou, and H. Zhang. LBMA: A novel locator based mobility support approach in named data networking. *China Communications*, 11(4):111–120, April 2014.

[82] Y. Rao, H. Zhou, D. Gao, H. Luo, and Y. Liu. Proactive caching for enhancing user-side mobility support in named data networking. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 37–42, July 2013.

[83] R. Ravindran, S. Lo, X. Zhang, and G. Wang. Supporting seamless mobility in named data networking. In *IEEE International Conference on Communications (ICC)*, pages 5854–5869, June 2012.

[84] G. Rossini and D. Rossi. A dive into the caching performance of content centric networking. In *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 105–109, September 2012.

[85] M. Safar, H. Farahat, and K. Mahdi. Analysis of dynamic social network: E-mail messages exchange network. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications &Amp; Services*, iiWAS '09, pages 41–48, 2009. ACM.

[86] M. Safar, H. Farahat, and K. Mahdi. Robustness of dynamic social networks. *Journal of Mobile Multimedia*, 6(3):243–262, September 2010.

[87] M. Safar, K. Mahdi, H. Farahat, S. Albehairy, A. Kassem, and K. Alenzi. Approximate cycles count in undirected graphs. *International Journal of Computational Intelligence Systems*, 7(2):305–311, 2014.

[88] M. Safar, I. Y. Sorkhoh, H. M. Farahat, and K. A. Mahdi. On maximizing the entropy of complex networks. In *The International Conference on Ambient Systems, Networks and Technologies (ANT)*, volume 5, pages 480 – 488, January 2011.

[89] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib. A survey on replica server placement algorithms for content delivery networks. *IEEE Communications Surveys & Tutorials*, 19(2):1002–1026, Secondquarter 2017.

[90] N. Samaan and A. Karmouch. A user centric mobility prediction approach based on spatial conceptual maps. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1413–1417, May 2005.

[91] V. Siris, X. Vasilakos, and G. Polyzos. Efficient proactive caching for supporting seamless mobility. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, June 2014.

[92] SQLite. SQLite: SQL database engine. http://www.sqlite.org/, 2015. Accessed on September 29, 2017.

[93] H. Tao, L. Jiangxu, L. Jiang, Z. Yunyong, and L. Yunjie. Mobility support for content source in content-centric networking. *China Communications*, 12(3):95–105, March 2015.

[94] F. Teraoka, Y. Yokote, and M. Tokoro. A network architecture providing host migration transparency. In *SIGCOMM Proceedings of the Conference on Communications architecture and protocols*, pages 209–220, September 1991.

[95] G. Tsirtsis, V. Park, and H. Soliman. Dual-stack mobile IPv4. RFC 5454, March 2009.

[96] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of the ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, pages 1–6, June 2012.

[97] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis. Proactive selective neighbor caching for enhancing mobility support in information-centric networks. In *Proceedings of the ICN workshop on Information-centric networking*, pages 61–66, August 2012.

[98] T. Woo, H. Park, S. Jung, and T. Kwon. Proactive neighbor pushing for enhancing provider mobility support in content-centric networking. In *International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 158–163, July 2014.

[99] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos. Caching and mobility support in a publish-subscribe internet architecture. *IEEE Communications Magazine*, 50(7):52–58, July 2012.

[100] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys and Tutorials*, 16(2):1024–1049, Second 2014.

[101] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM*, volume 2, pages 594–602, March 1996.

[102] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named data networking. *SIGCOMM Computer Communication Review*, 44(3):66–73, July 2014.

[103] Y. Zhang, H. Zhang, and L. Zhang. Kite: A mobility support scheme for ndn. In *Proceedings of the International Conference on Information-centric Networking*, pages 179–180, September 2014.

[104] M. F. Zhani, H. Elbiaze, and F. Kamoun. Analysis and prediction of real network traffic. *Journal of Networks*, 4(9), November 2009.

[105] Z. Zhou, X. Tan, H. Li, Z. Zhao, and D. Ma. MobiNDN: A mobility support architecture for NDN. In *Chinese Control Conference (CCC)*, pages 5515–5520, July 2014.

[106] Z. Zhu, A. Afanasyeva, and L. Zhang. A new perspective on mobility support. Technical Report NDN-0013, NDN, 2013.

[107] Z. Zhu, L. Zhang, and R. Wakikawa. A survey of mobility support in the internet. RFC 6301, July 2011.