Efficient Network Coding Protocols for Information-Centric Networks

Hirah Malik

université
PARIS-SACLAY

Efficient Network Coding Protocols for Information-Centric Networks

Protocoles de codage réseau efficaces pour les réseaux centrés sur l'information

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Réseaux, information et communications
Graduate school : Informatique et sciences du numérique
Référent : Faculté des Sciences d'Orsay

Thèse préparée dans les unités de recherche **Inria Saclay-Île-de-France** (Université Paris-Saclay, Inria) et **Laboratoire des signaux et systèmes** (Université Paris-Saclay, CNRS, CentraleSupélec), sous la direction de **Michel KIEFFER**, Professeur (Université Paris-Saclay), le co-encadrement de **Cédric ADJIH**, Chargé de recherche (Inria-Saclay), et le co-encadrement de **Claudio WEIDMANN**, Maître de conférences (CY Cergy Paris Université)

Thèse présentée et soutenue à Paris-Saclay, le 22 Nov. 2021, par

Hirah MALIK

Composition du jury

Steven MARTIN Professeur, Université Paris-Saclay	Président
Samia BOUZEFRANE Professeure, CNAM Paris	Rapportrice & Examinatrice
Jérôme LACAN Professeur, ISAE SUPAERO	Rapporteur & Examinateur
Vincent ROCA Chargé de recherche, Inria Grenoble	Examinateur
Marie-José MONTPETIT Professeure adjointe, Concordia University	Examinatrice
Michel KIEFFER Professeur, Université Paris-Saclay	Directeur de thèse

Thèse de doctorat

NNT: 2021UPASG096

Efficient Network Coding Protocols for Information-Centric Networks

Hirah MALIK

A thesis submitted in partial fulfillment
requirements for the degree of
Doctor of Philosophy
at Paris-Saclay University

Paris, November 22, 2021

Acknowledgments

ALHAMDULILLAH, all praises to ALLAH, the most merciful and beneficial, for all his blessings.

The road to completion of this thesis and the proceeding research was not smooth by any means. It was challenging to begin while the whole family was fighting an unfortunate health situation. It was equally challenging to continue after the birth of our second child right before the pandemic brought the entire world to a grinding halt. Working from home was excruciatingly difficult with two kids. It was a daily struggle, and at times I seriously thought of taking a break or even quit; however, the belief, encouragement, motivation, and constant support of many people kept me going, whom I would like to thank sincerely.

Firstly I want to thank my supervisors Prof. Michel Kieffer, Dr. Cedric Adjih, and Prof. Claudio Weidmann, for trusting me to be part of their incredible research team. Their invaluable guidance and adaptive support on every step during this Ph.D. were phenomenal.

Major credit to every achievement in my life goes to my mother, Mrs. Khalida Parveen, and this was no different. I am grateful to her for how she raised me with all the sacrifices she made throughout our lives. I always look up to her for her courage and perseverance. To my late father Mr. Adil Mehmood who would have been equally happy and proud.

I want to acknowledge and appreciate the crucial role of my mother-in-law, Mrs. Khizrat Bibi; without her help, none of it would have been possible. My late father in law Mr. Mohammad Shafi would have been proud to see me reach this incredible goal.

I share this accomplishment equally with my biggest supporter, my husband, Dr. Farrukh Malik, who believes in me more than I believe in myself. Thank you for pushing me constantly to take on challenges that I am hesitant to take up myself. I am blessed with all your love, continuous support, and persistence, mon amour. My amazing sons, Zayyan Malik and Issa Malik, for all the love, joys, and of course, the troubles and challenges you bring to my life.

My siblings, Mr. Fahad Adil, Mrs. Hajra Adil, and Mr. Usama Adil, for always cheering for me. A special thanks to my brothers, Mr. Naveed Bhatti & Mr. Waseem Bhatti, for their unconditional support at multiple levels.

I would also like to extend my gratitude to my friends Dr. Hiba Yousef, Dr. Anas Husseis, Ms. Abhiyadhatri Arige, Dr. Aakanksha Rana, and Dr. Quang Trung Luu for their continuous motivation, inspiration, and support whenever I needed it and to all my colleagues at INRIA for the excellent work environment.

Abstract

The amount of data exchanged over the Internet has grown dramatically over the past decades. The increasing number of users, connected devices, and the popularity of video content have surged the demand for new communication methods that can deal with the growing volume of data traffic. Information-Centric Networking (ICN) has been proposed as an alternative to traditional IP-based networks. In ICN, consumers request named content via Interest packets to the network and receive data as a response to their request from anywhere in network.

ICN allows in-network caching and naturally supports the use of multiple paths. Nevertheless, the maximum throughput can only be achieved if the content is requested over an optimal set of multicast trees. The computation of such multicast trees is hard to scale over large dynamic networks and requires coordination among network entities.

Network coding has been recently introduced in ICN to improve multi-path dissemination and caching of content without the need for coordination. The challenge in the case of network coding is to get independent coded content in response to multiple parallel Interests by one or several consumers. In this thesis, we analyze some previous works that integrate network coding and ICN and identify some key issues these works face. We introduce an efficient solution where clients add compact information to Interest packets in order to ensure linear independence of content in network-coded ICN.

This thesis proposes an architecture, MICN, that provides network coding on top of an Interest-based ICN implementation: Named Data Networking (NDN). The proposed architecture helps alleviate the issues faced by network coding-enabled ICN solutions presented in the past. A novel construction called MILIC (Multiple Interests for Linearly Independent Content) is introduced that imposes constraints on how the replies to Interests are coded, intending to get linearly independent contents in response to multiple Interests. Numerical analysis and simulations illustrate that the MILIC construction performs well with network-coded NDN, and the MICN protocol yields close to optimal throughput in some scenarios. The performance of MICN compares favorably to existing protocols. It shows significant benefits when considering the total number of transmitted packets in the network and in the case of lossy links. Several modified forwarding techniques integrated into the MICN protocol are proposed to optimize the network resource utilization while keeping a high throughput.

MILIC led us to consider the problem of constructing subsets of vectors from a given vector space, such that when drawing arbitrarily one vector from each

subset, the selected vectors are linearly independent. This thesis considers it as a mathematical problem and studies some alternative solutions to the MILIC construction. Finally, the thesis proves that a large family of solutions to this problem are equivalent to MILIC.

Keywords: Information-Centric Networking (ICN), Named Data Networking (NDN), Network Coding, Interest Forwarding Strategy

Résumé

Le volume de données échangées sur l'Internet a augmenté de façon spectaculaire au cours des dernières décennies. Le nombre croissant d'utilisateurs, d'appareils connectés et la demande croissante de contenus vidéo nécessitent de nouvelles méthodes de communication capables de gérer le volume croissant des données échangées. Les réseaux centrés sur l'information (Information Centric Networking, ICN) ont été proposés comme une alternative aux réseaux IP traditionnels. Dans les réseaux ICN, les clients demandent au réseau un contenu par son nom via des paquets d'« intérêt », et reçoivent des données en réponse à leurs demandes sans avoir à se soucier de l'emplacement du contenu dans le réseau.

Les réseaux ICN permettent la mise en cache dans le réseau et prennent naturellement en charge le multicast. Néanmoins, le débit maximal ne peut être atteint que si le contenu est demandé sur un ensemble optimal d'arbres multicast. Le calcul de ces arbres est difficile à mettre en œuvre sur de grands réseaux dynamiques et nécessite une coordination entre les entités du réseau.

Le codage réseau (Network Coding) a été récemment introduit dans les réseaux ICN afin d'améliorer la diffusion par l'utilisation de chemins multiples et la mise en cache des contenus sans qu'une coordination soit nécessaire. La difficulté dans le cas du codage réseau est d'obtenir des paquets codés linéairement indépendant en réponse à de multiples requêtes parallèles par un ou plusieurs clients. Dans cette thèse, nous analysons certains travaux antérieurs qui intègrent le codage réseau et les réseaux ICN et identifions limitations de ces approches. Nous proposons une solution efficace où les clients ajoutent des informations compactes aux paquets d'« intérêt » afin d'assurer l'indépendance linéaire des paquets reçus en réponse.

Cette thèse propose MICN, un protocole permettant une intégration efficace du codage réseau dans une implémentation d'ICN : Named Data Networking (NDN). L'architecture proposée permet de résoudre certains des problèmes rencontrés par les solutions ICN integrant des codage réseau présentées dans le passé. Une nouvelle construction appelée MILIC (Multiple Interests for Linearly Independent Content) est introduite. Elle impose des contraintes sur la façon dont les réponses aux paquets d' « intérêt » sont codées. Le but est d'obtenir des contenus linéairement indépendants en réponse à des intérêts multiples. Analyse théorique et simulations montrent que MILIC fonctionne bien avec le codage réseau pour NDN, et que le protocole MICN offre un débit proche de l'optimum dans divers scénarios. Les performances du protocole MICN se comparent favorablement aux protocoles existants. Il présente des avantages

significatifs lorsqu'on considère le nombre total de paquets transmis dans le réseau et dans le cas de liens pouvant subir des pertes. Plusieurs techniques de gestion de la diffusion des « intérêts » dans le protocole MICN sont proposées afin d'optimiser l'utilisation des ressources du réseau tout en conservant un débit élevé.

Lors de l'étude de MILIC nous avons considéré le problème de la construction de sous-ensembles de vecteurs dans un espace vectoriel donné, tels que lorsqu on choisit arbitrairement un vecteur de chaque sous-ensemble, les vecteurs sélectionnés sont linéairement indépendants. Cette thèse formalise ce problème et propose quelques solutions alternatives à la construction MILIC. Enfin, la thèse prouve qu'une large classe de solutions à ce problème est équivalente à MILIC.

Mots-clés: Réseaux centrés sur l'information, Named Data Networking (NDN), Codage réseau, Stratégies de relayage d'intérêts, Réseaux centrés contenu

Contents

List of Figures

List of Tables

Acronyms

CCN	Content Centric Networking
CDN	Content Distribution Networks
CEE	Cache Everything Everywhere
CR	Content Routers
CS	Content Store
DAG	Directed Acyclic Graph
DASH	Dynamic Adaptive Streaming over HTTP
DNS	Domain Name System
DONA	Data-Oriented Network Architecture
DRL	Deep Reinforcement Learning
ERF	Expected Rank-Based Forwarding
FEC	Forward Error Correction
FIB	Forwarding Information Base
FIFO	First-in-first-out
GDP	Gradually Decreasing Pipeline (size)
ICN	Information-Centric Networking
ICNRG	Information-Centric Networking Research Group
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol Version 6
IRTF	Internet Research Task Force
LCD	Leave Copy Down
MAB	Multi-Armed Bandit
MCTS	Monte-Carlo Tree Search
MICN	MILIC-ICN
MILIC	Multiple Interests for Linearly Independent Content
MPTCP	Multi-Path TCP
MTU	Maximum Transmission Unit
NC	Network Coding
NDN	Named Data Networking
NetInf	Network of Information
NFD	Network Forwarding Daemon
NLSR	Named-Data Link State Routing
NRS	Name Resolution System
NWCRG	Network Coding Research Group
OF	Optimistic Forwarding

PARC	Palo Alto Research Center
PF	Pessimistic Forwarding
PIT	Pending Interest Table
PPO	Proximal Policy Optimization
PRLNC	Prioritized Random Linear Network Coding
PS	Parallel Strategy
PSIRP	Publish-Subscribe Internet Routing Paradigm
pub/sub	publish/ subscribe
RF	Robust Forwarding
RL	Reinforcement Learning
RLNC	Random Linear Network Coding
RPF	Replication Factor-based Forwarding
SELIT	Sets Ensuring Linearly Independent Transversals
SPOC	Secure Practical Network Coding
SQF	Shorter Queue-based Forwarding
TBF	Threshold-based Forwarding
TCP	Transport Control Protocol
UCB	Upper Confidence Bound
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VoD	Video on Demand
WWW	World Wide Web

1

Introduction

1.1 Background

Development and advancement in technology have revolutionized our world and daily lives. The Internet has significantly impacted people's lives and has transformed the world into a global village. Over the past few decades, incredible tools and resources have been developed, allowing access to information at our fingertips. Easier connectivity and access to information have made content the center of our lives. Consumers have access to content anywhere in the world, with just a click, in a blink of an eye. The consumers are interested more in the content itself rather than its location in the network. The current Internet architecture, however, works on the principle that attributes the content to its location.

Information-Centric Networking (ICN) has been proposed to shift the Internet paradigm from *location-specific* to *content-specific*. ICN removes location from the attributes of the content, consequently allowing it to be cached anywhere in the network. ICN identifies and routes content based on its name, hence shifting the importance from *where* to *what*.

Some ICN architectures allow the consumers to request content based on its name via an Interest packet. The Interest is routed in the network until it reaches a cache that stores a copy of the requested content. The content is routed back to the requesting consumers using the reverse path of the Interest.

Since ICN removes end-to-end communication, it naturally supports utilizing multiple faces and facilitates in-network caching. ICN supports seamless consumer mobility since no location information is involved, and communication is not restricted between two endpoints. A content request may be satisfied by any closest cached copy. Each ICN packet is signed and encrypted, unlike

traditional communication that encrypts the channel between two endpoints. This provides additional mobility support.

Another ICN benefit comes from the fact that nowadays client devices, *e.g.,* smartphones, laptops, tablets, *etc.,* are equipped with multiple network interfaces, *e.g.,* Wi-Fi, 4G/5G, and Bluetooth. Each one of them can be used to retrieve content. However, the point-to-point nature of traditional networking requires establishing a session among endpoints. Simultaneous use of the available interfaces is not generally supported. MPTCP [2], a recent advancement, has multi-path support, but it requires multiple sessions for each source. ICN, however, provides intrinsic support for multi-path usage by allowing clients to forward their Interests to all available faces.

Overall the primary goal of ICN from its conception is to facilitate access to content by embedding the content-centric nature of client requests in the network architecture. Nevertheless, ICN does not instantly solve all the network challenges by itself, and in some cases, its operation can be tuned or further improved.

Considering the currently dominant Internet usage trends: most Internet traffic is generated by data-intensive applications such as video content consumption. In such applications, customer satisfaction is mostly based on the amount of time it requires to get back the content and the stability of content retrieval (*e.g.,* video quality). When several clients are interested in a large content, the ICN network faces issues that can degrade its performance. First of which is bottlenecks that can limit the flow of the content to the clients. Caching in the network [3] can help resolve this issue. However, it is only helpful if the Interests of multiple clients are timed one after the other so the content received in response to the first client's Interests can serve the other clients.

Additionally, with several clients requesting, the network faces the problem of limited bandwidth on the links between the clients and sources. Another vital issue that degrades ICN's performance, specifically in terms of delay, is packet losses. In case of packet loss, a client retransmits the Interest for the lost segment after the original Interest has expired. It induces a delay in retrieval and is problematic, especially for applications that require low latency, *e.g.,* video streaming applications. Finally, the optimal usage of multiple available paths is still difficult with the classical ICN, especially for multiple clients and sources scenarios.

1.2 Motivation

As described previously, there is a balance between the natural benefits of ICN and the limitations encountered in some scenarios. On the one hand, ICN inherently supports multi-path content retrieval, and it has been observed to increase the throughput [4]. The clients can retrieve content faster by aggregating the bandwidth of multiple interfaces. On the other hand, robust content dissemination is a challenge, and performance hits limitations described previously. In fact, the optimal throughput can only be achieved in multicast scenarios if the content is requested over an optimal set of multicast trees [5]. Finding the optimal multicast trees in large-scale and dynamic networks has scalability issues, especially with link failures and losses. It is also complicated in ICN due to its

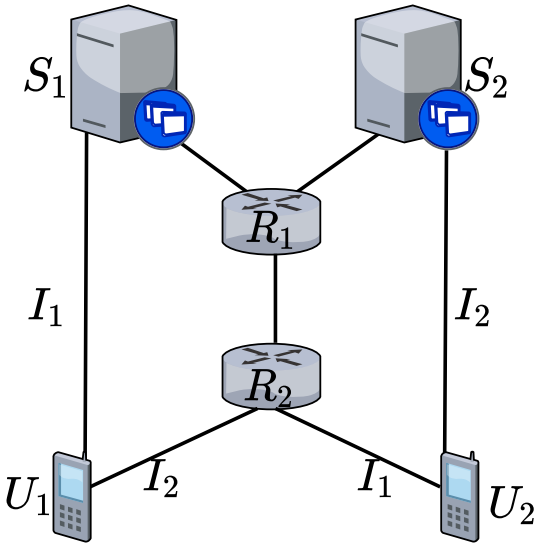


Figure 1.1 – Example of clients retrieving content in ICN over multiple paths

distributed and volatile content caching infrastructure. Complete knowledge of the network and coordination between several network entities is required.

To illustrate this last point, Figure 1.1 presents a simple example of clients U_1 and U_2 requesting some content in ICN. Both clients want a content consisting of two segments x_1 and x_2. They send Interests I_1 and I_2 for the two segments over their two available paths. We assume that client U_1 forwards Interest I_1 on the direct link to source S_1 and U_2 forwards Interest I_2 on the direct link to the source S_2. The other Interests of each client go through the link $R_1 - R_2$. Observe that the link $R_1 - R_2$ is a bottleneck and can only deliver one packet at a time. So the client U_1 will have to wait for the content x_1 to be delivered to U_2 before getting back the segment x_2 (considering client U_2's Interest I_1 was received and forwarded first).

If, however, the clients can coordinate and send the request for the same segments on the middle link, they can utilize the network capacity more efficiently. For example, if both nodes send the Interest I_1 on the link $R_1 - R_2$ they can achieve maximum throughput. However, it is only possible with coordination among the network nodes and knowledge of the network topology. Thus, an interesting question is how to operate ICN in an uncoordinated but efficient manner? This thesis focuses on studying network coding as a solution to efficiently use ICN for multi-path content retrieval.

Network coding has been proposed to solve ICN's problem when downloading large content in a multi-party network [6]. We will explain the benefits and challenges of this combination in the following section. The main idea of network coding is to allow network nodes to combine packets in the network, unlike routing, where the packets are only forwarded.

In classical ICN, clients sequentially request each of the data segments of content individually. However, with network coding, the clients request coded segments of the content with no particular order. Coded segments are generated by combining the source segments of the content. The network nodes, including the sources and the intermediate routers, can perform network coding operations

on the content segments and send coded segments in Data packets as a response. The Data packets also carry the encoding vectors of the coded segments that bring information about the encoding.

A request for coded segments of the content is satisfied by any coded segment instead of one specific segment, reducing the content request's granularity. Since any coded packets can respond to a coded content request, it allows multiple users to take advantage of others' requests without the need for coordination. Take, for example, the scenario in Figure 1.1. If the clients U_1 and U_2 send a request for coded segments instead of specific segments, one coded packet can respond to the request of both clients, allowing them to utilize the bottleneck link $R_1 - R_2$ efficiently.

Network coding enables the clients' requests to be sent over multiple faces without the need to forward them over a set of optimal multicast trees. The mixing of segments within the network increases the diversity of content in the network. The increased content diversity increases the cache hit rate. Network coding enables ICN to utilize multiple paths efficiently and consequently increases the throughput of the ICN network in data-intensive applications.

Along with the throughput gains, network coding can help ICN to improve resilience to losses. It can help ICN to recover from both Interest and Data packet losses. In the case of classical ICN, one Interest requests a specific segment. If the Interest for that segment or the Data packet is lost, the clients must re-request the segment. The client would have to wait until the corresponding Interest expires before re-sending the Interest in the network. However, with network coding, if a coded packet or an Interest is lost, another Interest or coded packet can be used instead. With network coding, the clients can proactively send a few more Interests in case of lossy network conditions.

A client requesting coded content can decode the original content once it has received enough linearly independent coded packets. Linearly independent coded packets are required for the clients to perform decoding operations. However, ensuring that clients get back a decodable set of coded content is not straightforward. Since ICN allows caching in the network and any coded segment can respond to Interest requesting coded segments, there is a high probability that the client may get back the same coded segment again from neighboring nodes. Several solutions have been proposed in the literature to ensure retrieval of a decodable set of coded segments, including sending the encoding vectors of all the received coded packets. The previous attempts have some shortcomings in terms of balancing between retrieving a decodable set of content and retrieving the content at the maximum possible rate. One open question is how to improve them to take full advantage of network coding in ICN?

1.3 Thesis Objectives

Motivated by the throughput benefits that network coding can bring to ICN, the main objective of this thesis is to propose a protocol that efficiently integrates network coding and ICN to overcome the shortcomings of the previous attempts.

We choose Named Data Networking (NDN), an Interest-based ICN architecture to integrate network coding with ICN. Ensuring that the clients get back a

decodable set of coded segments with the maximum possible rate is the primary design objective of the protocol. To retrieve a decodable set of coded content, finding the perfect naming scheme is imperative. The naming scheme includes the content names in the ICN packets and the associated meta-information in options fields. It should allow the Interests to advertise what is required at the client to decode content to the network. We aim to design a naming scheme with network coding that takes into account the principles of NDN and allows the processing of multiple parallel Interests in the network.

The second design objective of the protocol is to ensure maximum throughput in large content retrieval scenarios. This design objective translates into having a forwarding strategy that ensures that enough Interests are forwarded in the network.

The third objective of this thesis is to optimize network resource utilization while ensuring maximum throughput. Network resource utility is improved by optimizing the number of Interest and Data packets sent in the network. This optimization is to ensure that the protocol meets its primary objective without needlessly flooding the network.

The final goal of the thesis is an investigation on generalizing the mechanisms introduced to achieve the previous objective: the construction of subsets such that when choosing one vector from each subset, the resulting set of vectors are linearly independent. It includes exploring alternate families that satisfy this property and ensuring their suitability for network coding.

1.4 Thesis Contributions

In order to design an architecture that integrates network coding and NDN, we start by studying the previous works. We analyze in detail the design choices of architectures integrating network coding and ICN in the literature.

The first contribution of this thesis presented in Chapter 3 provides a solution for compressing received content information in the Interests using a linear code(s) approach. Many network-coded ICN works proposed in the past add the received content information in the Interests to ensure that the client receives a decodable set of coded content. However, the information is not added compactly and incurs significant overhead in the Interest packets. Our solution helps to reduce this overhead.

The second contribution of this thesis is also presented in Chapter 3. We present a formal analysis of the forwarding strategies of NetCodCCN. NetCodCCN is an existing family of protocols that aims to achieve similar design goals as ours. Our formal analysis presents that their forwarding strategies are not always efficient and that protocol proofs are important.

The main contribution of this thesis is the MICN architecture which integrates network coding over NDN presented in Chapter 4. We propose a specific naming scheme called MILIC (Multiple Interests for Linearly Independent Content). MILIC adds an index in the Interest name. These indices represent predefined subsets of coding vectors (following a set of mathematical constraints). The response to an Interest with a MILIC index can only have coding vectors from the predefined subset. The subsets ensure that the coded packet for each index is linearly independent of the others. We define these subsets and provide

formal proofs that linear independence of content for different subsets is ensured. MICN ensures that each client receives a decodable set of coded content with maximum throughput.

The maximum throughput of MICN comes at the cost of flooding of Interest. A consequence of this flooding is redundant traffic in the network. The following contribution of this thesis is proposing dynamic forwarding strategies in Chapter 5 to reduce the flooding of Interests and consequently reduce the extra data traffic.

The final contribution of this thesis is studying families of subsets similar to MILIC. We formalize the problem of defining subsets such that if one element is chosen from each subset, the resulting set of vectors is linearly independent. We study some general families and prove that they are equivalent to MILIC. Some examples of alternate algebraic solutions are also proposed.

1.5 Thesis Outline

The thesis is organized as follows

Chapter 2 highlights and overviews the fundamental properties of ICN in general and of NDN in particular. It presents the background and overview of the existing work performed in the content of NDN and network coding.

Chapter 3 analyses the design choices made in some of the pre-existing work in the context of our primary design goals.

Chapter 4 presents MICN, an architecture that integrates network coding and NDN to achieve maximum throughput for large data retrieval. The chapter presents a detailed architecture of MICN, the results, and the benefits of MICN compared to NDN and one of the pre-existing solutions. Moreover, the chapter also observes the flooding of Interests as a side effect of the maximum throughput that MICN achieves and presents an initial optimization mechanism to solve it.

Chapter 5 presents some modified forwarding techniques to counter the flooding problem introduced in Chapter 4.

Chapter 6 presents finding subsets that give linearly independent transversals as a formal mathematical problem SELIT problem. We prove that a large number of families of construction that are SELIT solutions are equivalent to our construction MILIC (introduced in Chapter 4).

Chapter 7 concludes this thesis and presents some future perspectives and promising directions for network-coded NDN architectures and MICN.

Some ideas and figures presented in the thesis have appeared previously in the following publications:

Journal Papers

(J1) Hirah Malik, Cédric Adjih, Claudio Weidmann, and Michel Kieffer. MICN: a Network Coding Protocol for ICN with Multiple Distinct Interests per Generation. *Computer Networks*, 187:107816, 2021.[7]

Conference/Workshop Papers

(C2) Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. Analysis of the Properties of NetcodICN Protocols. In *CORES 2020 – 5ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Lyon, France, September 2020.[8]

(C3) Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. On the Problem of Finding "Sets Ensuring Linearly Independent Transversals"(SELIT), and its Application to Network Coding. In *9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, pages 1–6. IEEE, 2020.[9]

2

State of the Art

2.1 Overview

Content distribution has become the primary task for today's Internet. Internet usage for content retrieval has increased many folds in the past decade and is continuously growing. According to CISCO's forecast, video traffic will be accounting for 79 percent of total mobile data traffic by 2022 [10].

The communication network's traditional paradigm has some drawbacks, especially when dealing with large-scale content distribution because of the point-to-point nature of communications and location dependence. In IP-based networks a connection between the requester and the source is established. Once a secure connection is established between the two endpoints, communication

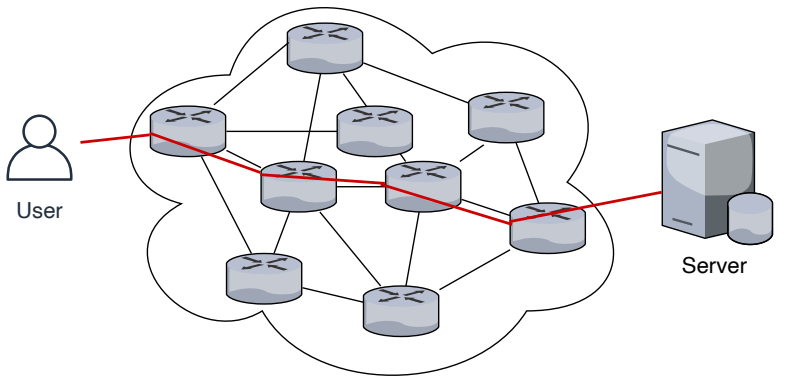


Figure 2.1 – Content retrieval in traditional IP networks

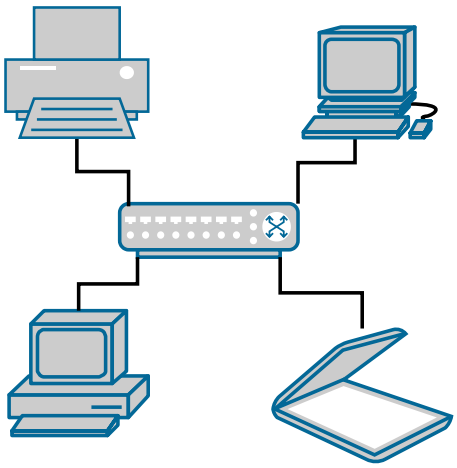


Figure 2.2 – Initial concept of communication among computers

takes place over this connection. The increasing interest in content, the evolution of consumer behaviors, and the amount of data being shared over the Internet (from smart devices, connected homes, IoT devices) have prompted a shift in this paradigm. Newer and faster Internet architectures are being explored for efficient content sharing. Consumer behavior has triggered the research around content-based networks because consumers care about the content itself and not about its origin.

Information-Centric Networking (ICN) is a novel approach introduced to make *content* the center of the network. It decouples the content from its location and allows content retrieval from anywhere in the network from a mere name. One challenge of the ICN architectures is to perform efficiently in multi-party (multi-source, multi-client) scenarios. Coordination among different network entities is required to overcome limitations such as content duplication, bottlenecks, *etc.*

An approach to address this shortcoming is to use coding of the content, *e.g.,* channel coding. The sources encode the original content using, *e.g.,* FEC erasure codes, raptor codes, *etc.*, and the clients can retrieve a subset of the coded packets generated by the source to decode the original content. However, it has been presented that allowing intermediate nodes to perform coding operations can further help the performance of content retrieval [11]. Therefore, network coding has been proposed as an alternative approach; moreover, some studies have demonstrated the benefits of network coding in ICN networks.

In this chapter, we present an overview of the concepts of ICN. We formally present the content retrieval process on the Internet and introduce content-centric networks and their advantages in the current Internet world. We introduce ICN and its popular implementations, focusing on Interest-based implementations. Within the Interest-based architectures, we focus mainly on NDN and describe all the principal elements and algorithms of content retrieval in NDN.

Later in this chapter, we introduce a few shortcomings in terms of fast content retrieval in classical NDN networks and detail some of the methods that have been used to overcome them. We focus primarily on network coding-based solutions. We discuss the previously published work in this domain. Finally, a

Figure 2.3 – Current Internet content delivery network

discussion on the forwarding techniques in network coding-based NDN networks is presented.

2.2 Motivation of Information/Content Centric Networks

Current Internet communication relies on the IP protocol [12] that assumes a host-centric networking model. Host-centric communication models enable communication between well-defined locations. This model originates from the very early communication networks and has survived their evolution. From the invention of electric telegraphs to the primary telephone network, the goal was to connect two hosts. The initial network started with dedicated cables to connect the endpoints. Then with the increase of users, a concept of switches was introduced to connect several entities.

The early computers allowed clients to use the machine's computational power remotely, and there was little data transfer between devices. Like telephone networks, the initial concept of communication among computers included a model involving two machines, and the amount of data shared was small.

The User Datagram Protocol (UDP), transmission control protocol and IP protocol (TCP/IP) were introduced to transport Data packets across a wide range of networks giving rise to inter-networking and the Internet. The introduction of the World Wide Web (WWW) for information-sharing and hyperlinking helped transform internet to a universal tool to access content. Over the last two decades, the Internet has become the prime source of information, *e.g.,* news websites, blogs, forums, and social media websites. It has also become the prime source of entertainment with video on demand (VoD), social media, online music and is the center of many businesses with e-commerce and online shopping.

The Internet architecture can be seen as a set $\mathcal{N}$ of nodes including content provides or sources $\mathcal{S}$, a set of content consumers or clients $\mathcal{U}$ and routers $\mathcal{R}$

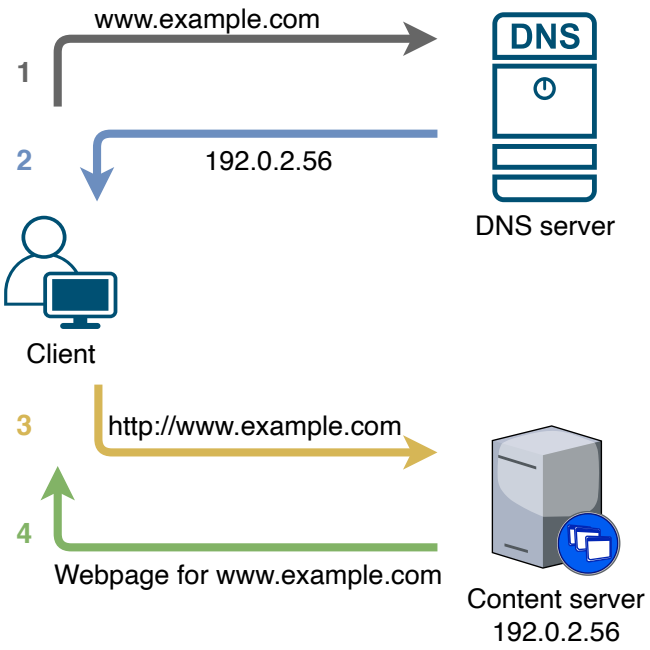


Figure 2.4 – Overview of content retrieval over Internet

that forwards copies of the packets as received. The operation of the Internet involves assigning a specific address to each of the network entities. Content retrieval over the Internet involves sending copies of the content object from a particular server to a requesting client.

The clients requesting certain content should know the server's location on the internet with the desired content, *i.e.,* the server's IP address. A specific name resolution system resolves the name into an IP address, *e.g.,* Domain Name System (DNS) [13, 14]. The DNS works like a phone book that manages the mapping between domain names and IP addresses. DNS translates the domain name that a client enters in its browser to an IP address.

The client can then use different methods to request the content from the content server. For example, with the current Internet architecture, the clients use Hypertext Transfer Protocol (HTTP) GET message [15] to send their content request to the IP address of the content server. There are other methods for the client to show interest in certain content, *e.g.,* subscribing to a PUSH [16] service. The content server then sends the content in HTTP response messages.

The amount of information shared over the Internet, and the use of this information has increased many folds. The effectiveness of the transfer of this information is critical in today's world. A substantial amount of research is devoted to improving the efficiency of communication architectures. However, the communication still depends on a client-server architecture.

The use of the Internet is evolving from a mean to connect a client to a server, to a universal tool of information access. The number of devices connected to IP networks is expected to be more than three times the global population by 2023 [17]. There will be 29.3 billion networked devices by 2023, up from 18.4 billion in 2018. These devices range from mobile phones, tablets to handheld games, smartwatches, and smart home appliances. In particular, the devices with video access can have a multiplier effect on traffic [17].

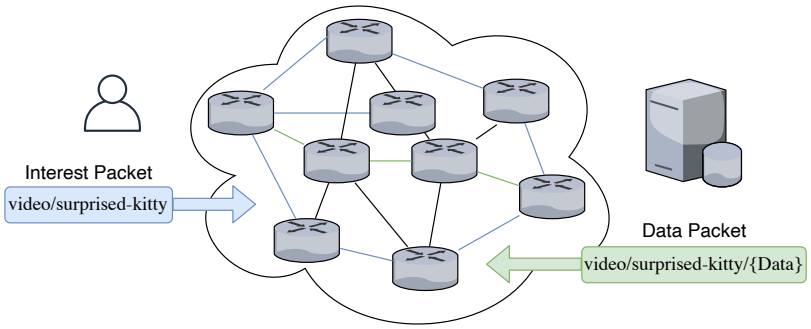


Figure 2.5 – Content retrieval in CCN/NDN

A wide range of overlay solutions for the existing Internet architecture have been proposed in the past, *e.g.,* peer-to-peer (P2P) [18], content delivery networks (CDNs) [19] to meet the increased demand. The increase in data traffic has also motivated network architectures that are different from the classical network paradigms. Among them, Information-Centric Networking addresses the issue of content delivery and mobility by introducing the concept of named content objects.

2.3 Information-Centric Networking

The explosion of video-based content consumption resulted in research of new approaches that can cater to the demand. Van Jacobson introduced the idea of shifting the communication paradigm towards a content-centric approach. He proposed that *content is the king* and stated:

> "If you don't care where you're getting the data—if all that matters to you is what the data is, not where it comes from—then all of this memory that has to be in the network as buffering in order to manage the multiplexing suddenly becomes a viable source of data.[20]"

This concept introduced the idea of remodeling or reengineering the Internet architecture. The perspective of focusing on *what* rather than *where* led to the notion of data-oriented networks, commonly know as Information-Centric Networks (ICN). ICN has been proposed as an alternative to the traditional point-to-point communication and shifts the paradigm towards a data-centric approach [21, 22]. The ICN principle is based on naming contents instead of their locations; it allows caching [3] at intermediate nodes to reduce the content delivery time.

ICN aims to better deal with a large amount of content by addressing caching, scalability, mobility, and security by focusing on the content in challenging communication scenarios [23]. Multiple network architectures based on ICN have been proposed in the literature. A few of the well known ICN architectures are listed below:

- Data-Oriented Network Architecture (DONA) [24]

- Network of Information (NetInf) [25]
- Publish-Subscribe Internet Routing Paradigm (PSIRP) [26, 27]
- Content-Centric Networking (CCN) [28]
- Named Data Networking (NDN) [29].

2.3.1 Data-Oriented Network Architecture

Data-Oriented Network Architecture (DONA) [24] proposed in 2007 is one of the first ICN-based architectures that proposed redesigning the Internet naming and name resolution. DONA introduced the concept of naming the content objects and giving globally unique names to content objects. The naming in DONA is organized around *principals*. Each data object name is associated with its principal, and the principal chooses the granularity of naming. The clients request the data by sending a *Find* message with the name. The Find message is forwarded until it reaches a copy of the content, which is sent back following the route of the find message. Each content object is associated with a public-private key pair. A client can verify that the data came from the principal by checking that the public key hashes to the principal.

2.3.2 Network of Information

The Network of Information NetInf [25] proposed in 2008 is an ICN architecture that effectively combines the elements of pub/sub and named content networks. NetInf allows named-based retrieval of the content, where clients request content by its name. A routing protocol advertises data object names and populates the routing tables of Content Routers (CRs). The subscribers send a GET message that is propagated hop-by-hop towards the publisher or a cache with a copy. It also supports the pub/sub architecture where clients subscribe to publishers based on the content they publish. A Name Resolution System (NRS) is used to map object names to locators that can be used to reach the corresponding information object. A publisher makes an information object available by sending a PUBLISH message with its locator to the local NRS. When a subscriber is interested in an information object, it can send a GET message to its local NRS, which routes the request to the publisher to bring back content in a DATA message.

2.3.3 Publish-Subscribe Internet Routing Paradigm

Publish-Subscribe Internet Routing Paradigm (PSIRP) [26, 27] proposed in 2012 tries to redesign Internet architecture from a publish/subscribe (pub/sub) point of view. In pub/sub architecture content sources are the *publishers* and clients are the *subscribers*. Sources publish what they have and can send, and receivers/clients subscribe to the publishers based on what they want to receive. In principle, a receiver will only receive what it has subscribed to. Additionally, each content object has a unique name for which the subscribers send special Interests. The Interests are matched to the published content by a *rendezvous system*.

2.3.4 Interest-Based ICN Architectures CCN/NDN

Content-Centric Networking (CCN) and Named Data Networking (NDN) are two of the most popular ICN architectures based on identifying content by *names*. The basic framework of CCN and NDN is a pull-based mechanism where the clients send *Interest* packets that contain the name of the requested content. These Interest packets are routed based on their names. A node holding a copy of the requested content replies to the Interest with the content in a *Data* packet.

Van Jacobson *et al.* originally proposed the Content-Centric Network (CCN) [28] architecture as a project initiated by Palo Alto Research Center (PARC). Named-data networking (NDN) [29] is an enhanced version of the CCN architecture and refers to the NSF-funded Future Internet Architecture project, a collaborative project that included PARC. The NDN project initially used CCNx as its code-base; after a few years of combined research, NDN and CCN parted ways. A research group ICNRG (Information-Centric Networking Research Group) in IRTF, provides a platform for ICN research ideas. They have many informational and experimental RFC documents on the ICN research topics such as review of research challenges, CCNx semantics [30, 31, 32] and others.

Although the primary mechanism is similar, there are some architectural differences between the two [33]. In this thesis, we focus on the NDN protocol suite. We use the NDN architecture and implement its core semantics in our simulator developed in Python. The implementations and simulations are on top of the basic NDN architecture, and we compare the results with the basic NDN.

In the next section, we present in detail the basic concepts and elements of the NDN architecture [1, 34].

2.4 Named Data Networking

Named Data Networking (NDN) is one of the Interest-based ICN architectures. Consider an NDN network consisting of a set $\mathcal{N}$ of nodes that can be *sources* that generate content, intermediate nodes or *caching routers* or *client*s that request content. A node can have any of these roles at a given time. Each node $r \in \mathcal{N}$ is connected in the network through a set of faces $\mathcal{F}_r$. The term *face* is a generalization for a network interface and corresponds to the communication channel that NDN implementation uses for packet forwarding. A good description of the detail of the NDN semantics and implementation is available in the NFD Developer's Guide [34].

2.4.1 Communication Packets

Communication in NDN is *consumer-driven,* with two basic types of communication packets: *Interest* and *Data* packets.

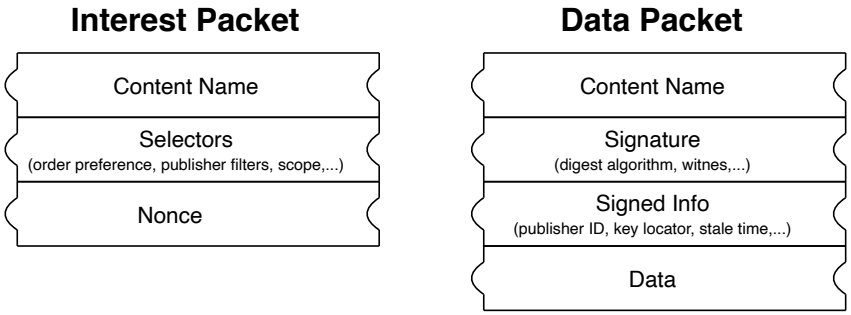


Figure 2.6 – Interest and Data packets in NDN architecture (From [1])

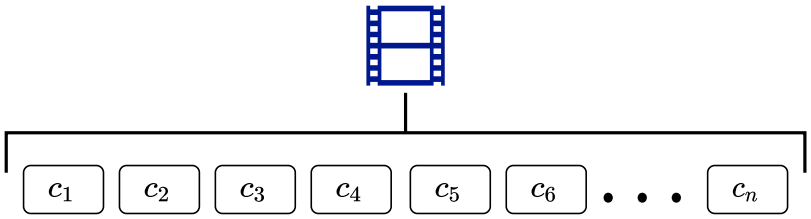


Figure 2.7 – Content segmentation in NDN

Interest Packets

Each content object in NDN is uniquely named. To receive content, a client sends the name of the content in an Interest packet. The Interest packet, as the name indicates, advertises a client's interest in a content by carrying the content's name[1]. The Interest is forwarded in the network hop by hop until it reaches a node holding a copy of the content with the requested name.

Data Packets

When the Interest packet reaches a source or cache that stores a *copy* of the requested content, the content is sent back in a Data Packet.

Content Naming and Segmentation

Each content object is identified by a name in NDN. Since every operation in the NDN network is based on names, naming is an essential aspect of NDN. The NDN naming is indifferent to network, *i.e.,* the names do not define an address. Each content object is named differently and independently by applications. Content is named hierarchically, which helps represent a relationship between data chunks and allows assisting the routing. The simplest form of the NDN names is of the following form `provider/video/surprised-kitty/`i that is the identifier of the content. Richer semantics with name prefixes, selectors, *etc.*, are also available at [35].

If the content object is large, it may be partitioned into smaller segments to fit into the Maximum Transmission Unit (MTU), and more generally may be transmitted through various use of NDN semantics (for instance [36]). A content object can be viewed as a set of data segments $C = [c_1, c_2, ..., c_n]$.

[1] along with potential options/selectors [34]

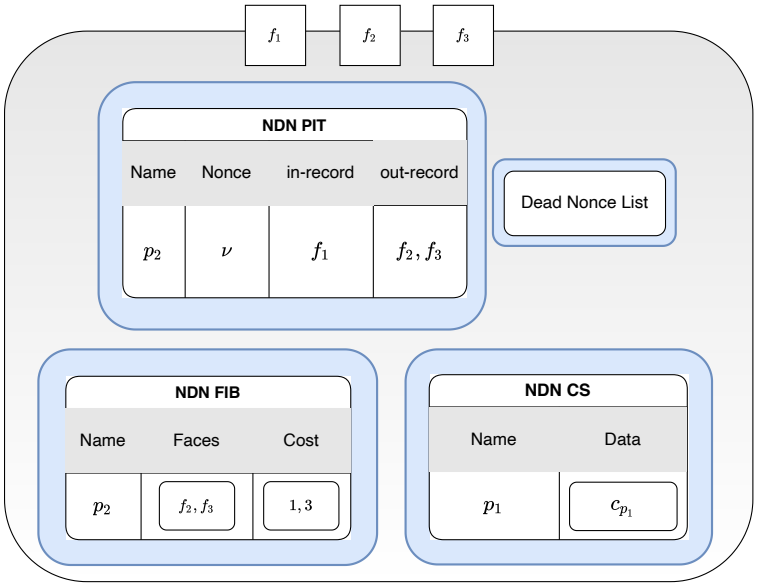


Figure 2.8 – Main components of an NDN-node

A client request can request any content by sending its name in an Interest packet. In the simplest mode of operations for large content objects, the clients send Interests with the content name and a segment identifier i. For example, the Interest `provider/video/surprised-kitty/`i is requesting the i^{th} segment of the content. The clients can deterministically construct the names of the subsequent Interest by incrementing the segment identifier. Each Interest also carries a random identifier, called *nonce,* that helps prevent Interest forwarding loops [34].

To retrieve the content C with n segments, a client has to send at least n Interests. The client sends more Interests in case of losses in the network. Each Interest has an expiration time associated with it. A client considers an Interest expired if no reply is received for it at the expiration time and can issue a new Interest.

The Data packets carry the same name as the Interest along with the content. Both Interest and Data packets carry the name of the content, but there is no information regarding the client or source.

2.4.2 Data Structures

The content retrieval in NDN is facilitated by data structures that store information necessary for routing and forwarding. Each NDN node has a *Pending Interest Table* (PIT), a *Forwarding Information Base* (FIB), and a *Content Store* (CS) for the transport of the named content in the network [28, 34] as shown in Figure 2.8.

PIT

The Pending Interest Table (PIT) keeps a record of pending Interests forwarded by the node that are not yet satisfied. PIT stores the information about faces where an Interest arrived and where it was forwarded so that the Data packet

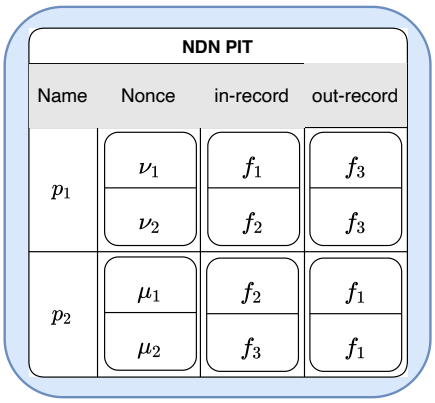


Figure 2.9 – NDN-PIT

can be routed back to the requester. PIT is a collection of PIT entries. A PIT entry is associated with each Interest that arrives at a node and is forwarded.

As represented in Figure 2.9, each PIT entry is identified by a name that is the name prefix of the content requested in an Interest. Each PIT entry has an associated in-record list and an out-record list. The in-record list stores the face(s) where Interests arrive (in-faces) and the out-record list stores the face(s) where they have been forwarded (out-faces). The information stored in the in-record list and the out-record list helps the nodes route the content back to the requesting node by storing the reverse link for the Data packet. Each PIT entry also has a nonce list to record the nonce(s) associated with the Interests that it has forwarded. The nonce information helps a node in loop detection. The node updates a PIT when an Interest is received or when the content for an Interest arrives, and in case of an Interest expiration.

FIB

The Forwarding Information Base (FIB) stores routing information used to forward Interest packets toward potential sources of matching content. Much like the PIT, the FIB is also a collection of entries, as represented in Figure 2.10. A name or prefix for some content identifies each entry. Each entry has a set of next-hop links that can lead to a potential source(s) for the prefix. A routing cost for each next-hop link is also associated. The FIB can be populated in several ways manually or automatically. The FIB population might be complex due to the fact that global information about the network can be necessary. A node can populate its FIB by self-learning, *i.e.,* keeping the information of the Interests it forwarded in the past. A routing protocol may also populate the FIB. Routing and forwarding strategies to efficiently perform the name-based routing are presented in [37] and [38].

Content Store (CS)

Content in NDN is independent of location and can be stored anywhere in the network. Therefore, each NDN node is not just a router that routes packets but is also equipped with cache memory to store content. This cache memory is called the content store (CS). Each content is in the content store is identified by it name as represented in Figure 2.11.

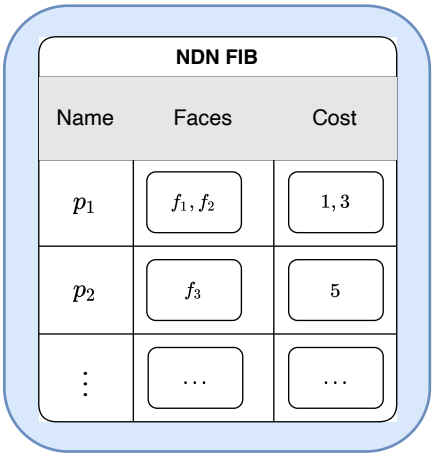


Figure 2.10 – NDN-FIB

Any node in the network can cache the content that it forwards downstream. Caching the content can help nodes to reply to future Interests. Consequently, content is stored in source nodes and in caching routers [34]. Before forwarding any Interests, the nodes first consult their CS to check if a copy of the requested content is stored.

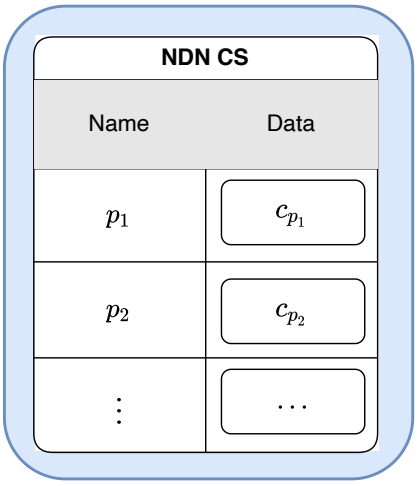


Figure 2.11 – NDN-CS

The data might be cached in the CS for as long as possible. The cache, however, has a limited capacity and the data has to be evicted after some time. For any content in the cache, the nodes implement caching policies for caching and eviction decisions. Caching policies keep track of the content requests and then decide on storing or removing content based on either popularity or time it has been stored in the cache. Caching is one of the important research topics in NDN communities. Some popular caching strategies are LRU (Least Recently Used), CEE (Cache Everything Everywhere), LCD (Leave Copy Down). Some of the caching approaches in NDN are detailed in [39].

Dead Nonce List

Dead Nonce List is another data structure in NDN that keeps a record of the nonces that a node has processed (forwarded or replied) in the near past. This data structure is introduced mainly to avoid looping Interests. The Dead Nonce List supplements the PIT in loop detection. The Interests are usually short-lived and expire, and the PIT removes expired Interests along with the nonce. The reason for storing these nonces is detecting possible looping Interests that may arrive sometime after the Interest has expired in the PIT. As the name indicates, the dead Nonce list stores a list of nonces that are *dead* from the PIT.

2.4.3 NDN Interest Processing

An NDN node processes every received Interest to decide whether it can reply to this Interest, forward it or ignore it as represented in Figure 2.12. If a forwarding decision is taken, the Interest processing decides the face(s) where to forward the Interest. The incoming Interest processing involves several steps. The first of which is the detection of looping Interests by checking the Dead Nonce List. If the nonce of the received Interest exists in the Dead Nonce List, it is considered looping and immediately dropped; this is called Interest suppression. Then the node consults the PIT for duplicate nonces, and the Interest is dropped if the same nonce is found in the PIT.

The nodes then verify if a similar Interest, *i.e.,* an Interest requesting the same content, is already pending in the PIT. An existing entry with the same name indicates that the node has already forwarded a similar Interest. An existing entry also confirms that the requested content is not available in the cache. If a similar entry exists, the node only updates the entry, and the new Interest is not forwarded since content for it is already expected; this process is called Interest merging. The pre-existing entry is updated by appending to their respective lists, the incoming face, and the Interest's nonce. Also, the Interest expiration time for the entry is updated. Storing the incoming face allows the node to store the path to send back content, and storing the nonce helps to perform loop detection.

If a similar entry is absent, the node then verifies if the requested content is available in the cache. A content store lookup checks if content with the same name as the Interest is stored in the cache. If there is a *cache hit, i.e.*, the requested content is available in the cache, the node responds to the Interest with a Data packet. The Data packet is sent on the incoming face of the Interest.

In the case of the unavailability of requested content or *cache miss*, the node decides to forward the Interest. The node can choose to forward the Interest on one or several faces. A forwarding strategy takes the forwarding decision, *i.e.,* the number of faces and the exact faces where the Interest is forwarded. The forwarding strategy uses the faces stored in the FIB for the prefix to select among the available faces.

Then the node creates a PIT entry to store the in-face(s), out-face(s), and the nonce.

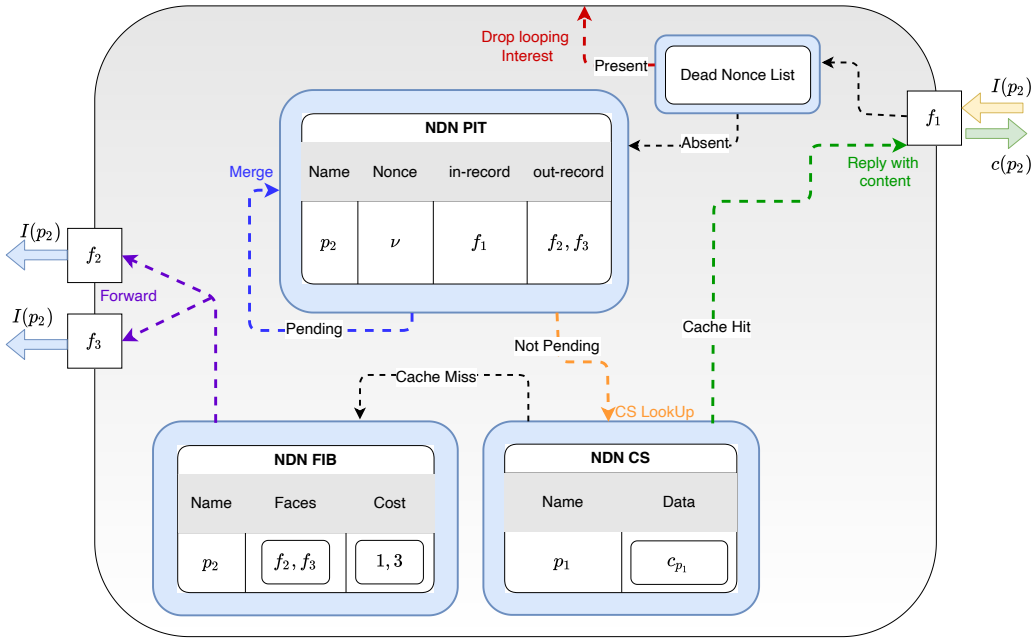


Figure 2.12 – NDN Interest processing

2.4.4 NDN Data Processing

Once a Data packet is received from upstream, the node verifies if it had previously forwarded a corresponding Interest. If not, the content is considered unsolicited and not forwarded anywhere, as represented in Figure 2.13. However, the node can choose to cache it based on its caching strategy. If an Interest for the received content is present in the PIT, the information stored in its PIT-entry (in-record) is used to find the path to the client downstream. If the Interest was received from more than one face, a copy is sent to each of those faces. Once the content is routed back to the requesting node using the information in the PIT, the node deletes the corresponding PIT entry. The node also decides whether the content should be cached locally [34].

2.4.5 Forwarding in NDN

In NDN, the forwarding decision is taken by a forwarding strategy. The strategy decides where an Interest should be forwarded; how many faces, and which exact faces. Some of the default NDN forwarding strategies are listed below [34].

- **Best Route Strategy** chooses to forward the Interest to the face with the lowest cost of reaching a source.
- **Multicast Strategy** forwards the Interests to all the available faces.
- **Client Control Strategy** allows the clients to decide where to forward the Interest.
- **Load Balance Strategy** forwards on the available faces alternatively to balance the load.

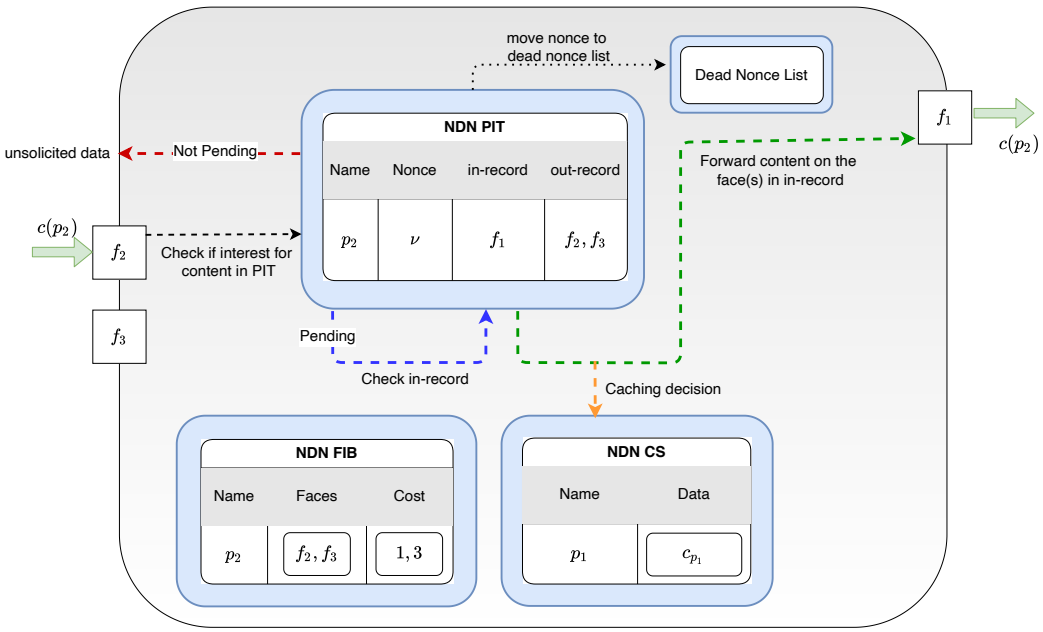


Figure 2.13 – NDN data processing

2.4.6 Multi-path Content Delivery in NDN

Devices nowadays come with multiple network interfaces that can be used to retrieve content, *e.g.*, Wi-Fi, 4G/5G. Traditional networking requires establishing a transport session among endpoints. Using multiple paths is possible in IP, *e.g.,* with MPTCP [2], but it requires associating multiple paths to multiple addresses. A detailed survey of the multi-path delivery over the Internet is presented in [40].

In contrast to traditional networking, NDN inherently enables the use of multiple interfaces. NDN does not require establishing a connection, and the clients and intermediate nodes can choose to use multiple paths. Multi-path content delivery is an important research topic in NDN networks. Rossini *et al.* [4] analyze the content delivery in NDN networks, focusing on multi-path forwarding strategies. Gomes *et al.* [41] propose load balancing mechanisms for NDN to utilize multiple paths. They show an improvement in utilizing multiple paths compared to typical single path strategies.

Nevertheless, NDN cannot always perform optimally in a dynamic network with multiple clients and sources when using multiple paths. There is a one-to-one mapping between Interests and content, *i.e.,* each Interest corresponds to a specific chunk of content. Due to the one-to-one mapping, the use of multiple paths does not bring appreciable benefits in NDN. If a client sends an Interest I_{p_1} requesting segment 1 of content with prefix p over multiple faces, it will receive only multiple copies of the same content at the client. In a scenario with clients requesting the same content over multiple faces, some coordination is required to take advantage of multiple paths.

Raptor codes [42] or packet level FEC codes have been proposed to improve ICN content delivery in multi-source and multi-client scenarios. Several works demonstrate their advantages in multi-client scenarios. Anastasiades *et al.* [43] propose RC-NDN an NDN architecture with Raptor codes. Evaluations show

that RC-NDN outperforms the original NDN significantly.

Montpetit *et al.* [6] introduced network coding over ICN as a solution to the coordination approach to use multiple paths in ICN efficiently. Numerous works have been proposed using network coding over different ICN architectures. Before detailing the application of network coding over ICN, we present the basics of network coding.

2.5 Network Coding

Network coding (NC) is a modern communication paradigm [11]. It allows the network nodes to perform operations on the packets instead of just routing them. Traditional network nodes only forward the received packets. However, network coding suggests that mixing the incoming information flows is possible. It is not necessary to keep them independent, and merging or mixing them can improve network performance. Network coding can benefit communication networks in terms of throughput, security, forward error correction (FEC), resilience to link failures or losses. One of the significant benefits specifically for NDN networks is the increased Data packet diversity in multi-path scenarios.

Network coding combines packets in the network. Nodes receiving different packets can combine them by performing different reversible operations. These operations can be XOR, addition, or linear combinations [44]. These network-coded packets are then forwarded to the network. The receiving nodes perform the decoding operations to recover the initial information flows.

The benefits of network coding were initially presented by Ahlswede *et al.* [11] in a wired network. The example butterfly scenario in Figure 2.14 illustrates the benefits of network coding in a wired network in [11]. An analogous scenario without coding was presented in Section 1.2 and Figure 1.1. There are two sources S_1 and S_2, and two clients U_1 and U_2. All links have same capacity 1. As with [11], but unlike our later assumption of a packet network, this description assumes a synchronous communication network where bits (symbols) are individually transmitted on links. The sources generate bits x_1 and x_2.

In a network with standard routers, the middle link should transmit one of x_1 and x_2 ; without the loss of generality, the Figure 2.14 assumes it is x_1. The client U_2 receives both bits simultaneously. Client U_1 receives x_1 from both its links. For client U_1, the node R_1 has to resend later the bit x_2 along the path $R_1 - R_2$ for instance.

However, with network coding the router R_1 can take x_1 and x_2 and generate a third bit $x_1 + x_2$ by combining them, *e.g.,* by XORing the two bits. $x_1 + x_2$ is now sent over the bottleneck link (see Figure 2.14b). The client U_1 receives $\{x_1, x_1 + x_2\}$ and the client U_2 receives $\{x_2, x_1 + x_2\}$. Both U_1 and U_2 can solve the simple system of two equations to retrieve the original bits. With network coding, both clients receive the content simultaneously.

The example in Figure 2.14 illustrates the network coding benefits in terms of throughput and delay, compared to classical routing. Network coding can also be helpful in case of losses, the coded packets can help retrieve lost packets. Gkantsidis *et al.* [45] demonstrate that download time for coded large scale content is significantly lower in comparison to forwarding unencoded information.

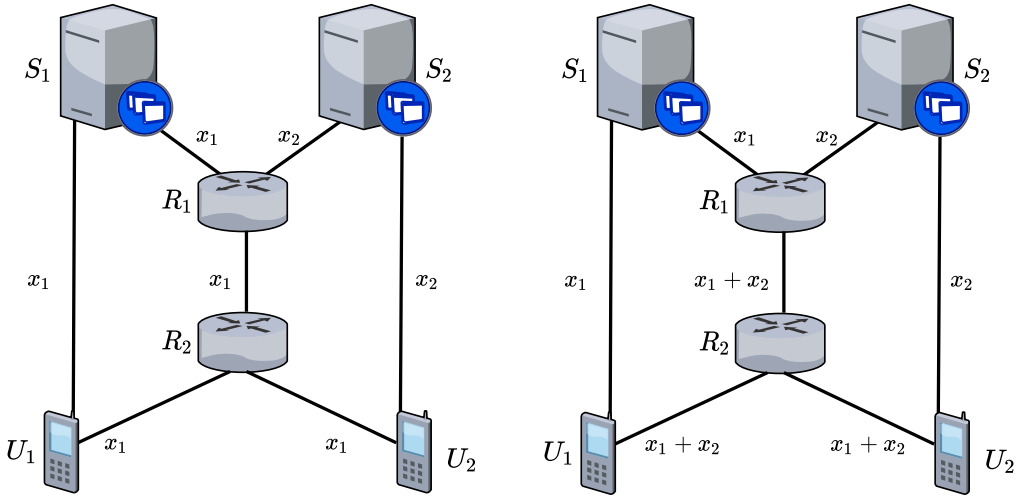


Figure 2.14 – Example network for network coding advantages

Additionally they demonstrate that network coding improves the download time further compared to coding at the server only. A few other works presenting details on network coding are [46, 47, 48, 49].

2.5.1 Random Linear Network Coding RLNC

One of the commonly used methods of network coding is performing linear operations on packets in the network [44]. Packets are considered to be vectors over a given Galois Field $\mathbb{F}_q$. $\mathbb{F}_q$ is a Galois field of q elements and $\mathbb{F}_q^*= \mathbb{F}_q \setminus \{0\}$. In RLNC [50, 51], nodes can independently perform random linear combinations of the packets, and the coding coefficients are chosen randomly. The original data is divided into segments of equal size and then linearly combined by choosing scalar coding coefficients drawn from a finite field. *Coded* segments are a linear combination of source segments seen as vectors of the Galois field $\mathbb{F}_q$. For instance $Q_1 = \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_n P_n$ where $P_1, P_2, \ldots, P_n$ are the source segments and $(\alpha_i)_{i\in\{1,\ldots,n\}}$ are the coding coefficients. The coded segments are of the same size as the source segments. Once enough linearly independent coded segments are received, the receiving nodes can decode and recover the source segments by solving the linear system of equations. An important feature of RLNC is that it allows re-encoding without the need of decoding. Therefore, the network nodes can combine coded segments to create more coded segments, increasing the network's data diversity (recoding). The re-encoded segments also have the same size as coded or source segments.

2.6 Network Coding and ICN

Network coding and ICN both inherently tend to address the content delivery and focus on improving content distribution over the network. Network coding and ICN can work jointly to exploit network capacity better (by exploiting

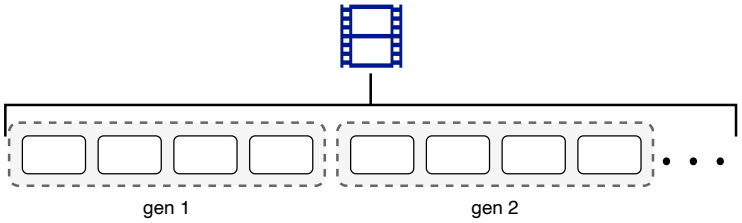


Figure 2.15 – Content segmentation in network coding-based NDN

caching, multi-path delivery, *etc.*). The idea of applying network coding over ICN was first introduced in [6] to take advantage of ICN and network coding's inherent features to improve content delivery. The idea was to counter the challenges ICN faces when dealing with large content retrieval presented in [52]. They described the network coding implementation over a Pub/Sub ICN architecture [27] and an Interest-based architecture (CCN) with Network Coding for CCN (NC3N) [6]. Motivated by the advantages that network coding can bring to ICN [6, 53], several works have been presented combining network coding and ICN. There are efforts in standardization on these topics. For instance, at IRTF, the Network Coding Research Group (NWCRG) focuses on coding for the network, including network coding and sliding window coding [54]. The Information-Centric Networking Research Group (ICNRG) focuses on ICN. In particular, an ongoing work [55] presents a combination of ICN and network coding.

2.6.1 Network-Coding and Interest-Based ICNs

In this thesis, we focus on Interest-based ICN architectures and implementation of network coding over them. We begin by explaining the basic semantics of network-coded NDN and the challenges faced when incorporating the two.

In a classical network coding scenario, the original content C is partitioned into G smaller groups of segments, called *generations* $C = [c_1, c_2, ..., c_G]$. All generations of a content have same number of equally sized elements. Network coding is only allowed among the segments of the same generation. This helps reducing the decoding complexity and overhead.

With linear coding [44] segments from a given generation may be linearly combined within a source or at any intermediate node in the network. The coefficients in $\mathbb{F}_q$ of the linear combination form the *encoding vector* [56] of each coded segment. The encoding vector is associated to each coded segment. In RLNC the coefficients are chosen randomly. In the NDN context, the source nodes and caching routers may store original or coded segments.

In the most general application of network coding, the clients would not request a specific segment; instead, they request a coded segment. The client nodes send Interests requesting coded segments of a generation of the content. The name carried by Interest packets is adapted to indicate that coded segments are expected (*e.g.*, by setting a flag that indicates the retrieval of coded segments [6]). A client can send the Interest $\widehat{I}_g$ for the prefix p, indicating that coded segments from generation g are requested. The segment id in the NDN Interest is now replaced with the generation id.

The Data packets that clients receive in response to these Interests are encoded and contain linear combinations of source segments. A coded data segment contains the coded content and information of how the content was coded by carrying the encoding vector that indicates the weight of each of the source segments.

A client requires at least n linearly independent coded segments to be able to decode a generation of size n. The pull-based request-response mechanism of NDN allows one Interest to bring back one content segment. A client node has to send at least n Interests to be able to get n coded packets.

When receiving the Interest $\widehat{I}_g$ for coded segments, if the corresponding cache is not empty, a node can respond with a coded packet stored in its cache or can re-encode several cached packets to generate a new coded combination. The one-to-one mapping between Interest and content no longer exists in this network-coded scenario since we are assuming that any coded packet of the generation g can be a reply to such Interest.

The challenge when using network coding with NDN is to ensure retrieval of *innovative* (linearly independent) content in response to multiple parallel Interests by one or several clients. NDN allows caching and nodes can send the packets stored in their cache to respond to the Interest. In the case of network coding, this can be a problem: the same coded segment can respond to several Interests requesting coded content. Imagine a client that sends a request for a coded content and gets back the first coded segment. When sending the Interests for the next segment may get back the same coded segment from its neighboring node. However, in general, this can happen anywhere in the network. Two Interests from a client sent on different links may still end up bringing redundant content.

Another problem when requesting network-coded content is that when these multiple identical Interests are sent in parallel (pipelined), the nodes consider them to be similar Interests. With classical NDN processing, they are not forwarded further as content from another similar Interest is already expected. So in order to take advantage of the "one Interest, multiple possible responses concept", the Interest and content processing needs to be adapted in network coding enabled NDN.

2.6.2 Network-Coded NDN Architectures

NC3N proposed in [6] had Interests that include the information of coded segments available at the client sending the Interest. This approach is similar to the one proposed for TCP by Sundararajan *et al.* [57]. Later, the network-coded NDN schemes proposed in [58, 59, 60] work with the same principle. The encoding vectors of all the received coded segments are sent in the Interests. The encoding vectors help the nodes to decide either to generate a coded segment that will be innovative for the requesting node or forward the Interest to their next-hop neighbors. Wu *et al.* proposed the coding cache [58, 59] that stores coded content in the nodes of the network. By increasing the diversity, they are able to improve the cache hit rate. Similarly, Lei *et al.* [60] improved the cache diversity and an increased cache hit rate, and improved network transmission efficiency, a metric for the amount of useful data (linearly independent in this

case) transferred in the network.

Nevertheless, adding the coding coefficients of the received content [6, 58, 59, 60] introduces an overhead in the Interest packets. This overhead increases with the number of coded segments received by the requesting node (See 3.1). The size of the overhead can be limited by keeping the generation size small. However, these approaches also introduce a delay as the client node needs to wait to receive the replies for previously sent Interests before it can issue Interests for more coded segments to ensure retrieval of linearly independent content with each Interest. In fact, to always ensure linear independence client can only have one pending Interest and will operate in a stop-and-go manner.

Llorca *et al.*[61] proposed a caching-aided, network-coded scheme for multicast scenarios. They presented a framework to optimize the transport using random linear network coding, multicasting and Zipf popularity distribution for content. However, an implementation solution for the proposed scheme in ICN is not presented.

Liu *et al.* [62] introduced an Interest coding and forwarding strategy that allows splitting and joining Interests (for the same content and generation) at intermediate nodes. The Interests request a subset of segments by indicating the number of required coded segments to get a decodable generation. This scheme implements a one Interest-multiple replies strategy, which is not compliant with the NDN principle of one Interest-one reply.

Zhang *et al.* [63] compared the approach of sending all the encoding vectors of the received content (precise matching) to rank-based matching, *i.e.*, sending only the client node's rank. They observed that rank-based matching achieves slightly lower performance but has much lower computation and communication overhead.

Liu *et al.* [64] proposed to add the number of desired coded segments as an additional field in the Interest. These Interests allow the node to request more than one coded segment with one Interest. Consequently, a responding node can send more linearly independent segments in response to this Interest if it can generate more than one. However, requesting more than one coded segment contradicts the NDN principle of one Interest-one reply.

Matsuzono *et al.* [65] proposed L4C2, a low-loss, low-latency, network coding-enabled video streaming protocol over CCN. In L4C2, nodes request uncoded content. Network-coded packets are requested only in case of Data packet losses. Bilal *et al.* [66] proposed an algebraic framework of linear network coding for CCN for efficient data dissemination.

Lal *et al.* [67] proposed a framework for cache management in ICN based on network coding. They introduced different measures to choose the ideal caching routers to increase the diversity and consequently the cache hit rate.

Saltarin *et al.* presented NetCodCCN [68] that addresses the shortcomings of previous approaches by sending undifferentiated Interests for coded segments of a generation. The client node implicitly states that it requires another coded segment by sending additional Interests for coded segments. The intermediate nodes that have previously sent coded segments keep track of the number of coded segments forwarded on each face and the rank of their CS. The node only replies to an Interest if the rank of its CS is larger than the number of coded segments it has sent on a particular face. Thanks to these precise forwarding

rules, NetCodCCN also supports the transmission and handling of multiple Interests at one time (pipelining) and allows nodes to request content more efficiently. With pipelining, a burst of Interests is sent first by a client. Each time content is received, a new Interest is sent. A non-obvious drawback of this approach is that unnecessary data traffic can still flow in the network even after the clients have received enough content to decode a generation.

Saltarin *et al.* [69] and Bourtsoulatze *et al.* [70] presented adaptive video streaming schemes implemented over network-coded ICN. Saltarin *et al.* implement DASH (dynamic adaptive streaming over HTTP) in NDN. It is based on a modified version of their work presented in [68] and adapted to NDN. The scalable video transmission scheme presented in [70] uses Prioritized Random Linear Network Coding (PRLNC) [71] for encoding the different layers. It proposes the use of Bloom filters [72] to handle multiple Interests and Data packets.

In Chapter 3, we analyze some design choices of the works that used network coding over NDN in the past. The in-depth analysis provides us with insights that help us make design choices for our solution that implements network coding over NDN. Chapter 4 of the thesis proposes our novel protocol MICN that enables efficient data retrieval over NDN. MICN solves some shortcomings of the aforementioned approaches. It avoids the overhead incurred by some approaches, reduces the delay, and improves retrieval time. Indeed, MICN allows the forwarding and processing of several Interests in parallel.

3

Overview of Design Choices for Network Coding with ICN

3.1 Introduction

Network coding was introduced to solve ICN's coordination problem as detailed in Chapter 2, that also presented some challenges when incorporating network coding and ICN and the solutions proposed in the past to address these challenges.

In this chapter, we present in detail some design issues and choices that we explored in our effort to combine network coding with Interest-based ICN in the most efficient manner. We begin by explaining the main goals and challenges of combining the two approaches.

One of the main goals of incorporating network coding into ICN is to allow fast retrieval of content, especially in a multiparty network with multiple sources and multiple clients requesting the same content. Since Interest-based ICN uses a pull-based request-response mechanism, network coding-based ICN should work on the same principles. Plain ICN allows one Interest to bring back one content segment. A client node has to send at least n Interests to be able to decode a generation of n segments. However, in a network-coded scenario, one Interest sent over different links can bring differently coded packets (as there is no one-to-one mapping between Interests and content).

One of the initial ideas is to increase the number of forwarded Interests in the network, *i.e.,* forwarding Interests for coded content to multiple available faces and forwarding multiple Interests to one face (pipelining). The former is not particularly helpful in classical ICN, since Interests with the same prefix are expected to bring identical content. With classical Interest processing, multiple

Interests can not reach the sources because of Interest merging or suppression. Interest suppression occurs if an Interest with the same prefix (Interest requesting the same content) as an Interest already forwarded by the node is received again. The node only records this Interest and does not forward it. This is done to stop the propagation of Interests that are expected to bring back identical content. However, with network coding, one Interest for coded content can be replied to with different coded packets, and parallel processing of these Interests inside the network is beneficial. Doing so requires some adaptation of the Interest forwarding mechanism of classical ICN.

In addition, to decode the requested n-segment content, no matter how many Interests are sent in the network, the client requires n linearly independent contents. So the fast retrieval of content relies on the assumption that the content packets being received are all linearly independent. To achieve fast retrieval, we identified three basic principles for ICN with network coding.

Principle 1 Forwarding enough Interests in the network,

Principle 2 Ensuring that each of the Interests brings linearly independent content.

Principle 3 Pipelining, *i.e.,* parallel processing of multiple Interests.

Several works in the literature have proposed solutions following these principles. Solutions presented in [58, 59, 60] add the encoding vectors of the received coded packets in Interests and this ensure generation of linearly independent content with each Interest. This provides a good solution to ensure linear independence (Principle 1); however, they do not implement nor can benefit from sending multiple Interests or pipelining (Principle 2 and 3). Additionally, a naming overhead is introduced in the Interests with this method.

As discussed previously, classical ICN semantics is at odds with Principle 1. Another family of protocols called NetCodCCN [68, 69, 73] sends undifferentiated Interests in the network. They modify Interest forwarding at nodes to ensure parallel processing of these undifferentiated Interests. These protocols ensure linear independence by keeping counters (detailed in Section 3.3.1) and rank information at the intermediate nodes. They achieve Principles 1, 2, and 3.

We analyze the proposed solutions and explore the design space for these principles. In this chapter, we have two contributions. In the first part of chapter we propose an improvement of the solution that relies on sending a description of the content already at the client. We propose an efficient way to compress the information on received content to reduce the overhead in the Interests. This information helps ensure linear independence with each Interest.

In the second part of the chapter, our contribution is a detailed analysis of the NetCodCCN protocol family that performs parallel processing of multiple undifferentiated Interests requesting coded content. We identify some design issues and their consequences and show that the forwarding techniques introduced in different variants of these protocols are not always efficient, despite

satisfying all the principles. We also present how the findings of this detailed analysis influence our design choices for MICN: a network-coded NDN protocol that we propose and present in Chapter 4.

3.2 Retrieval of Linearly Independent Content

One of the critical design issues in network-coded ICN is to get innovative (linearly independent) content with each Interest. In network-coded ICN, the clients send Interests requesting coded segments of content. Such Interests can be replied to with differently coded segments of the content. However, it is challenging to ensure that the response to each of these Interests brings useful segments (linearly independent segments). To ensure linear independence among the received coded segments, the clients may either choose to

a) advertise what they have already or

b) state what they need to decode a generation.

The former has been proposed in several previous solutions (including [58, 59, 60]). In this section, we review these solutions, their benefits, and their limitations. We also present a novel solution to efficiently compress the information in the Interests.

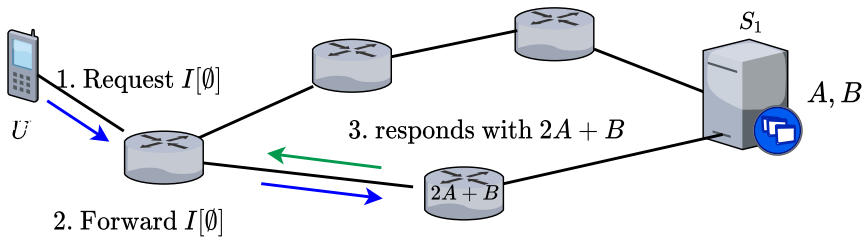
3.2.1 Network-Coded ICN with Vector Space in Interest

The client nodes can advertise what they have already received by sending the encoding vectors of the received segments in the Interests. They do so by adding a field with encoding vectors in the Interest represented by $I[EV]$.

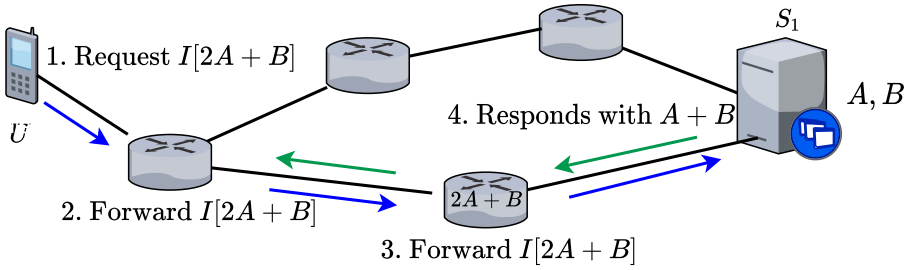
When the clients begin content retrieval, their Interests are sent with an empty encoding vector field $I[\emptyset]$, as they do not have any content. Any node in the network with the requested content generates a coded segment in response to the Interest $I[\emptyset]$. The coded segment is sent back in a Data packet. An example is presented in Figure 3.1, where the client requests a content with two source packets A and B. The first Interest $I[\emptyset]$ is answered with a coded packet $2A + B$ by the neighboring node.

The client issues a second Interest after receiving content for the first Interest, adding its encoding vector in the Interest. This Interest now carries information of the already received segments at the client. In Figure 3.1b the node receiving this Interest checks if it can generate a coded segment that will be linearly independent to the already received one. If not, it forwards the Interest to its next-hop neighbors. The Interest is forwarded until it reaches a cache with enough content to generate a linearly independent coded segment. This coded segment is guaranteed to increase the rank of the requesting node. This technique ensures that an innovative packet is sent as a response every time.

This approach provides an efficient solution to ensure linear independence with each Interest (Principle 2). Nevertheless, this approach introduces an overhead in the Interest packets that increases with each coded segment that the requesting node receives. The size of the overhead is limited by keeping



(a)



(b) Interest with encoding vector field responded with a coded vector linearly independent of the already received encoding vectors

Figure 3.1

the generation size small in some of the proposed work, *e.g.*, Lei *et al.* in [60] set the generation size 4 for their experiments. In the following sections, we focus on proposing a solution to compress the information of already received content at the client. Our contribution provides an effective way to reduce this overhead.

3.2.2 Vector Subspace Representation

We begin by introducing the concept of representing the content received at the client nodes as vector subspace. Assume that some client node has received k Data packets with linearly independent encoding vectors $v_i \in \mathbb{F}^n$, $i = 1, \ldots, k$. These row vectors form a basis of a k-dimensional vector subspace $\mathbb{V}$ of $\mathbb{F}^n$, with

$$\mathbb{V} = \text{span}\left\{v_1, v_2, ..., v_k\right\}.$$

For the next content to be linearly independent the next Interest packet should request some coded segment that does not belong to $\mathbb{V}$. The choice considered in [58, 59, 60] to represent $\mathbb{V}$ is to send $\{v_1, v_2, ..., v_k\}$ in the Interest, which requires $k \times n$ elements of $\mathbb{F}$ in the Interest name suffix.

3.2.3 Channel Coding Perspective

Alternatively, $\mathbb{V}$ can be seen as an (n, k) linear code over $\mathbb{F}$ considering a channel coding perspective. A $k \times n$ *generator matrix* G associated with this code is obtained by stacking the vectors v_i (as in [58, 59, 60]). Any element of $\mathbb{V}$ is obtained by linearly combining the rows of G. Then an efficient way to check whether a vector belongs to $\mathbb{V}$ is by considering an $(n - k) \times n$ *parity-check*

matrix H associated to G. The parity-check matrix is such that for all $v \in \mathbb{F}^n$,

$$s = vH^\top = 0 \text{ if and only if } v \in \mathbb{V}. \tag{3.1}$$

From (3.1) for any vector v to belong to the vector space $\mathbf{V}$, the syndrome of v relative to the parity-check matrix H must be equal to zero [74, pp. 86]. G and H are thus two equivalent representations of $\mathbb{V}$. Any generator matrix G can be put in the following *systematic* form

$$G_\mathrm{s} = \left[I_k | P\right] \tag{3.2}$$

by performing pivoting on rows of G followed by some permutation of the columns represented by the permutation π. In (3.2), I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix with elements in $\mathbb{F}$. If G_s has been obtained from G only by row pivoting, G_s and G represent the same vector space $\mathbb{V}$. This is no more the case when column permutations are necessary. The $(n-k) \times n$ parity check matrix associated with G_s is obtained as

$$H_\mathrm{s} = \left[P^\top | I_{n-k}\right]. \tag{3.3}$$

Again, if column permutations were required to put G in systematic form, H_s is not a parity check matrix of the code associated to $\mathbb{V}$. A parity-check matrix H for G is obtained by applying the inverse column permutation π^{-1} to the columns of H_s. To represent $\mathbb{V}$ using G or H, only the $k \times (n-k)$ matrix P is required, as well as the permutation function π.

3.2.4 Interests with Compressed Vector Space

We attempt to reduce the overhead in the Interests by using parity check matrices. The Interests sent for the content can be represented as $I(\mathbb{V})$, containing some function as a suffix to represent $\mathbb{V}$. From the definitions above, sending G when $k \leqslant n/2$ and H when $k > n/2$ provides a first reduction of the overhead in Interest packets compared to the approach considered in [58, 59, 60]. A single bit is necessary to indicate whether G or H is put in the Interest packet.

Then the alternate representation of H is sending the matrix P. The overhead in Interest packets with this approach is then $k \times (n-k)$ elements of $\mathbb{F}$ and at most $n \times \lceil \log_2 n \rceil$ bits to represent the permutation π or just n elements to represent the pivot columns. Consequently, when client nodes request network-coded segments of a generation, they may indifferently use $\mathbb{V}$, G, H, or P and the associated permutation function π in the Interest packet to indicate their received vector subspace. As a result, a node holding a vector v that is not in $\mathbb{V}$ can reply to the Interest.

Example 3.1. Consider content with generation size $n = 10$ over a Galois field $\mathbb{F} = GF(2^4)$[1]. A client has received 7 coded segments from the generation. The received packet information can be represented by the matrices G, H or P. The client can use any of these representations in the Interest $I(V)$ to ensure retrieval of linearly independent content as a response.

[1]For simplicity, we represent the field elements by $0, \ldots, 15$. The field was constructed using the irreducible polynomial

An example vector space of the client can be represented as V by stacking all the linearly independent coded segments received by the client.

$$V = \begin{pmatrix} 15 & 15 & 12 & 8 & 13 & 1 & 8 & 6 & 2 & 7 \\ 14 & 6 & 14 & 4 & 14 & 0 & 2 & 9 & 12 & 4 \\ 8 & 12 & 13 & 4 & 4 & 8 & 11 & 7 & 11 & 11 \\ 3 & 13 & 2 & 12 & 1 & 2 & 3 & 14 & 6 & 6 \\ 0 & 3 & 0 & 4 & 4 & 9 & 2 & 4 & 15 & 4 \\ 14 & 11 & 9 & 14 & 7 & 15 & 1 & 6 & 8 & 10 \\ 1 & 7 & 7 & 2 & 10 & 0 & 13 & 8 & 5 & 10 \end{pmatrix}$$

The $k \times n$ Generator matrix in (3.2) for the vector space V is given by

$$G_s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 14 & 11 & 7 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 6 & 9 & 3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 7 & 2 & 3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 10 & 5 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 7 & 10 & 13 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 7 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 9 \end{pmatrix}.$$

The resulting G matrix is obtained by some permutation of the columns represented by the permutation function

$$\pi = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The permutations π can, in fact, easily be represented by n indices to represent the pivot columns as $\pi^* = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 7 & 6 & 8 & 9 \end{pmatrix}$.

The generator matrix is in standard form $G_s = \left[I_k | P\right]$. The Generator matrix includes the $k \times (n-k)$ P matrix

$$P = \begin{pmatrix} 14 & 11 & 7 \\ 6 & 9 & 3 \\ 7 & 2 & 3 \\ 10 & 5 & 1 \\ 7 & 10 & 13 \\ 1 & 7 & 12 \\ 0 & 1 & 9 \end{pmatrix}$$

The related $(n-k) \times n$ parity check matrix 3.3 of the generator matrix is

$$H_s = \begin{pmatrix} 14 & 6 & 7 & 10 & 7 & 1 & 0 & 1 & 0 & 0 \\ 11 & 9 & 2 & 5 & 10 & 7 & 1 & 0 & 1 & 0 \\ 7 & 3 & 3 & 1 & 13 & 12 & 9 & 0 & 0 & 1 \end{pmatrix}$$

It is evident that the size of the parity check matrix H is much smaller than the Generator matrix G. And the P matrix is even smaller.

Example 3.1 presents the difference in the dimension of the matrices that can effectively represent the vector space received by a client. In order to compare the number of bits required to send each of these representations, we present Example 3.2.

Example 3.2. Considering a content with generation size $n = 16$ and a Galois field $\mathbb{F} = GF(2^8)$, the number of bits required to send the information about received coded segments using different approaches is given in Table 3.1.

No. of received coded segments k	G	H	P
1	128	1920	120
2	256	1792	224
8	1024	1024	512
9	1152	896	504
14	1792	256	224
15	1920	128	120

Table 3.1 – Number of bits[2] required to encode the received vector space of a client in Interests for content of generation size $n = 16$ from Galois field $\mathbb{F} = GF(2^8)$

It is evident that the overhead of sending G (as in [58, 59, 60]) increases linearly with k. This overhead can be decreased by sending the parity check matrix H after receiving segments $k > n/2$. The overhead introduced by the final approach (P matrix) is considerably lower even after considering the permutation function. The number of bits required to represent the vector space decreases with the number k of received coded packets.

3.2.5 Benefits and Limitation

Sending the vector space in the Interest packet ensures linear independence of content received with each distinct Interest. The parity check matrix method is an efficient way to compress the vector space representation to keep the overhead small, while still retrieving linearly independent content.However, this approach introduces some delay. The client node needs to wait to receive the reply for previously sent Interest to arrive, before it can issue another Interest requesting more coded segments. Multiple Interests carrying the same vector space issued together or sent over multiple faces may not bring similar benefits. The client can get back more than one packet for these multiple Interests. However, multiple coded packets generated in reply to the same Interest are not guaranteed to be innovative. There is a high probability of receiving linearly dependent content with these Interests.

Figure 3.2 presents an example scenario where multiple Interests with the same encoding vector fields are sent in the network. The initial Interest with

[2]The permutation is represented by a vector of pivots π^* for which $nlog_2[n] = 64$ bits max are required

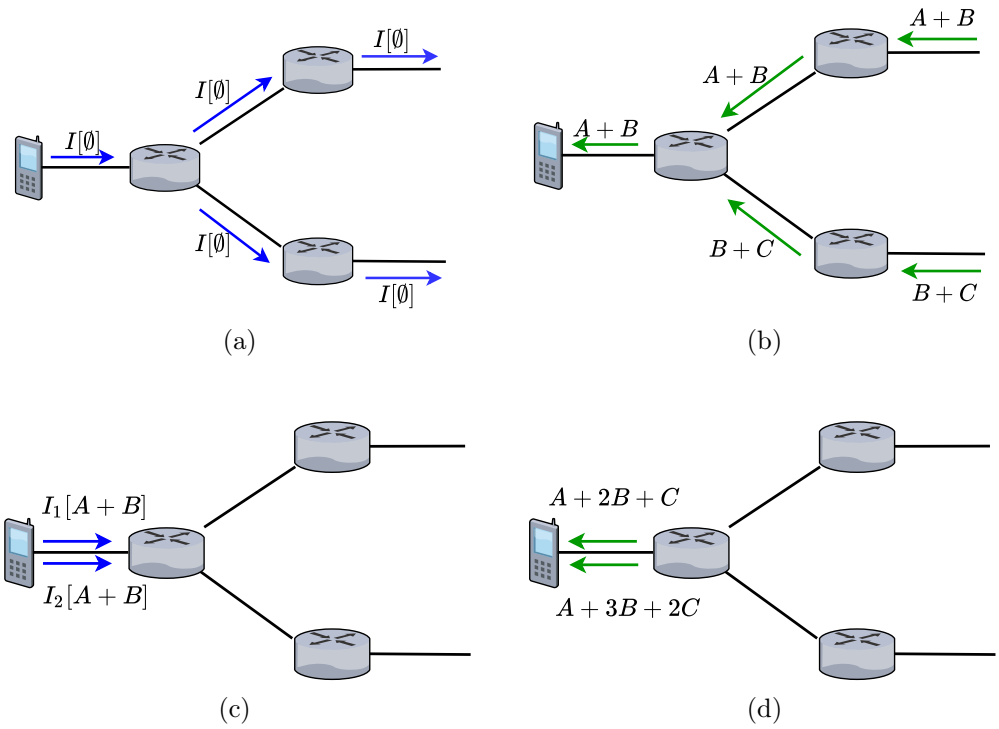


Figure 3.2 – Example scenario for multiple Interests with encoding vector field

an empty vector field brought two coded packets to the neighboring node of the client. One of them ($C = A + B$) was delivered to the client. The client then issues multiple Interests with the same encoding vector field, *e.g.,* two similar Interests in Figure 3.2. The neighboring node can reply to both Interests with differently coded packets $C_1 = A + 2B + C$ and $C_2 = A + 3B = 2C$. However, C, C_1 and C_2 are linearly dependent. This illustrates that sending multiple Interests is not always beneficial with this approach. Hence, one can not benefit from efficient forwarding or pipelining when using the approach of advertising received content.

An effective solution to this problem is to request content for multiple generations in parallel. Note that this approach has not been proposed in the past. Although it can overcome the delay over several generations, this kind of solution works only for problems where multiple generations of content are already available for download, or for applications where initially a complete download of the content can be done. However, such solutions are not well suited for scenarios where there is live streaming of content.

The shortcoming in terms of delay led us to explore the second method, where the client nodes state what is required to decode a generation instead of the vector space of what they have. The method is not detailed here but constitutes of of the major thesis contributions in Chapter 4. The main idea is to be able to send and process multiple Interests for network-coded data in the network. An alternate solution, MILIC, is presented in Section 4.3, that allows the node to implicitly state what is required to decode a generation. This is done by specifying constraints in the Interests for the content that can satisfy them. Interests with different constraints always recover linearly independent

content due to MILIC properties. The exact method is presented in Chapter 4. Pipelining of multiple Interests is possible with MILIC Interests.

3.3 Network Coding with Undifferentiated Interests

In Chapter 2 we presented some work from literature proposing network-coded ICN algorithms. The protocols NetCodCCN [68, 73] and NetCodNDN [69] (implemented on top of CCN and NDN, respectively) present a solution with undifferentiated Interests that can be pipelined in the network. These protocols are capable of reaching the capacity of the network or the maximum possible throughput. Although the capacity is shown to be reached on examples, we are not aware of formal proofs of this performance. The Interest forwarding algorithm is a critical aspect of these protocols, because their performance is mainly dependent on how parallel Interests reach the sources and bring back content.

Looking closely into these forwarding algorithms and their variants we found out that they have a flooding effect, *i.e.,* many Interests (more than required) are sent in the network. We reproduced the simulation results following closely the descriptions provided in [68]. The simulation results show that a lot of unnecessary data traffic is generated in the network due to the flooding.

In the following sections of this chapter, we analyze more formally the properties of several variants of forwarding algorithms of these protocols. For the rest of this chapter, we denote the family of protocols presented in [68, 69, 73] as *NetCodCCN*. The detailed analysis of NetCodCCN semantics and forwarding strategies contributed in some major design choices for the main contribution of this thesis (presented in Chapter 4).

3.3.1 NetCodCCN Semantics

NetCodCCN [68] is based on the idea that clients send several undifferentiated Interests, in reply to each of which linearly coded segments of the content will be sent, possibly from different sources. Receiving enough linearly independent coded segments will be sufficient to retrieve the original content. The core idea of these protocols is different from classical CCN/NDN forwarding. Instead of suppressing similar Interests, these protocols allow the nodes to receive and forward several identical Interests from a face. To keep track of the sent Interests and expected content, the nodes keep a count in the in-records and out-records. The NetCodCCN PITs store more information than classical CCN PITs, and accordingly, the forwarding algorithms are different.

We first present the main building blocks of the NetCodCCN nodes, followed by the algorithms for Interest and content forwarding. A NetCodCCN node has data structures differing from CCN/NDN nodes. Each data structure stores information per generation for each content prefix.

1. Each node has a PIT. The PIT maintains for each of its faces f:
   - r_f = the counter of pending Interests (the in-record);

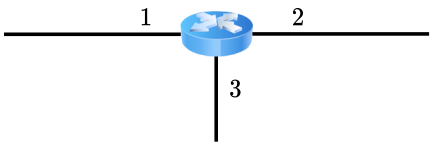


Figure 3.3 – NetCodCCN node (with three faces)

- s_f = the counter of the Interests that have been forwarded on the face (the out-record);
- σ_f = the number of coded segments that have been sent on the face;
- s_{all} = the total number of incoming Interests that required forwarding (from any face).

2. Each node has a CS. The NetCodCCN CS stores linearly independent coded segments $\{Q_1, Q_2, \ldots Q_\rho\}$. The number of these linearly independent segments defines the node rank ρ. The node rank is used for the decision of Interest forwarding and data response.

3. Each node has a Forwarding Information Base (FIB), denoted $\mathscr{F}$, which indicates the interfaces on which the Interest could be forwarded. The forwarding strategy determines dynamically on which subset of $\mathscr{F}$ they are actually forwarded.

Figure 3.4 shows the detailed structure of a NetCodCCN node with three faces represented in the simple topology in Figure 3.3.

Now we present the network operation of the NetCodCCN nodes.

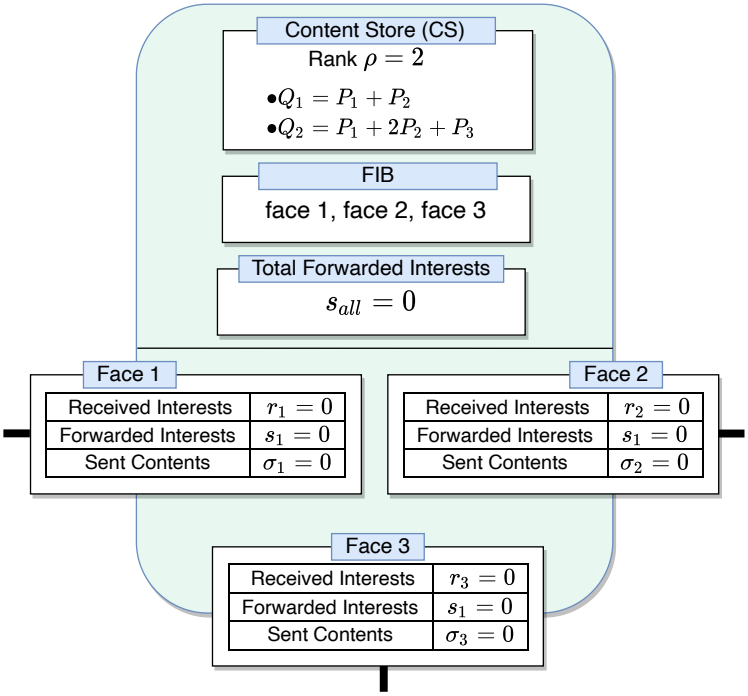


Figure 3.4 – Internal data structures of a NetCodCCN node

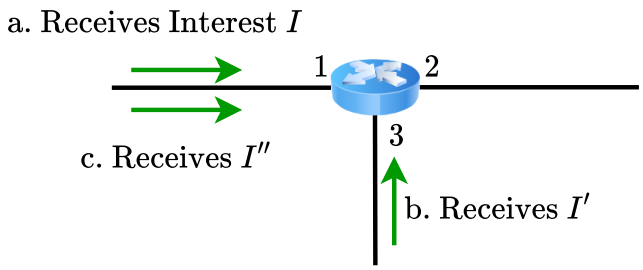


Figure 3.5 – Example forwarding scenario

Interest Processing

NetCodCCN uses a *Parallel Strategy* (PS). The parallel strategy is a multicast forwarding strategy that forwards Interests in parallel over all faces in the Forwarding Information Base (FIB). When receiving an Interest on a face f, the node processes this Interest before deciding whether to forward it. The node first checks its rank, *e.g.,* if it has full rank, it can decode all coded segments. If so, the node can act as a source and send a linear combination of packets. Otherwise, if $\sigma_f < \rho$, then the CS has more linearly independent combinations than it has already forwarded on the face f. In that case, the Interest can be immediately satisfied by sending a random linear combination of coded segments in the CS. Otherwise, the Interest is considered for forwarding.

Interest Forwarding Strategies

NetCodCCN protocols have undifferentiated Interests, *i.e.,* all Interests requesting coded segments of the same content appear similar. Similar Interests are merged or suppressed in normal CCN/NDN forwarding, as they are expected to bring the *same* content. However, since this is not the case for coded scenarios, a NetCodCCN node forwards similar Interests. A node decides if and where it will forward the Interest based on the information stored in its PIT and its rank. An Interest is never sent back on its receiving face.

We identify several variants and derivatives of NetCodCCN presented in [68, 69, 73]. They vary in their Interest forwarding decision algorithms. We present the forwarding variants in detail; the naming of the forwarding variants appearing here is ours. We consider the example scenario presented in Figure 3.5 with a NetCodCCN node with three faces to present these variants. The node receives undifferentiated Interests requesting coded segments of some content in the sequence a, b and c as represented in Figure 3.5. The node receives a first undifferentiated Interest I from face 1, then a second one I' from face 3, and afterward a third Interest I'' from its face 1 again. The Interests are represented as I, I' and I'' only to indicate that they are received from different faces at different times.

We consider that the CS of the node is initially empty. The node records all the information in its PIT. The node has to decide whether or not it will forward the Interest, based on the counters in its PIT. The forwarding decision is taken based on one of the following variants.

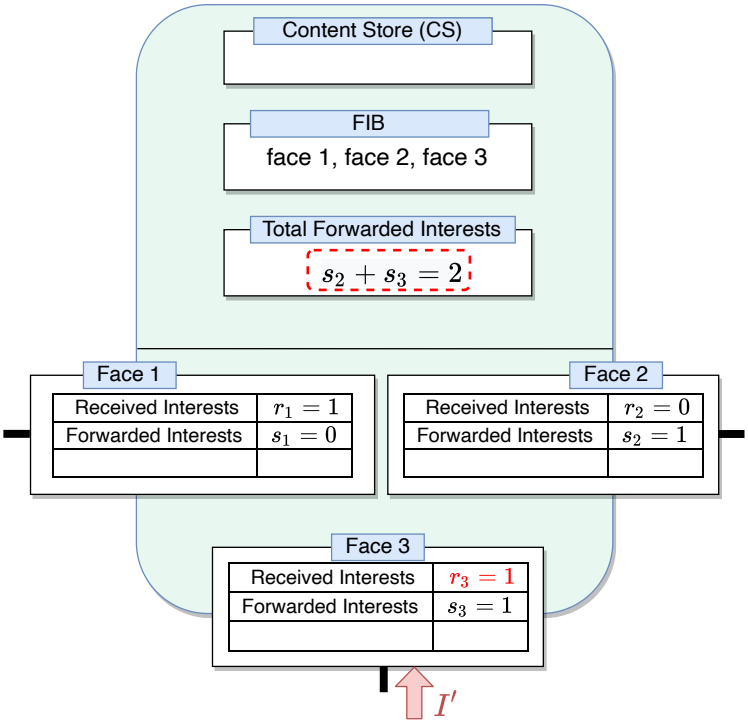


Figure 3.6 – Example scenario for Optimistic Forwarding

Optimistic Forwarding (OF)

Optimistic Forwarding assumes that all sent Interests on all faces will bring back content. Hence forwarding occurs only when the number of received Interests becomes greater than total Interests forwarded $r_f > \sum_{\varphi \in \mathcal{F}} s_\varphi$. The assumption here is that every face $f \in \mathcal{F}$ brings back innovative content. This is proposed in [69, p. 2190] and [73, p. 53]. The state of the node's data structures (counters) are presented in Figure 3.6 during the scenario from Figure 3.5.

- The first Interest is received at face 1, updating the received counter r_1 to 1. It is forwarded to the faces 2 and 3, incrementing their forwarding counters s_2 and s_3 to 1 each.
- When the second Interest arrives at face 3, its received counter r_3 updates to 1. However, the forwarding does not occur since the total forwarded Interests computed by OF is $\sum_{\varphi \in \mathcal{F}} s_\varphi = 2$, and the condition $r_f > \sum_{\varphi \in \mathcal{F}} s_\varphi$ is not met. Figure 3.6 presents the state of the node before the arrival of second Interest I'.
- The same happens when the third Interest I'' arrives at face 1, *i.e.*, the forwarding condition is not met, and the node does not forward the Interest. This is because the first Interest has been forwarded twice

So only one Interest will be forwarded out of three with optimistic forwarding.

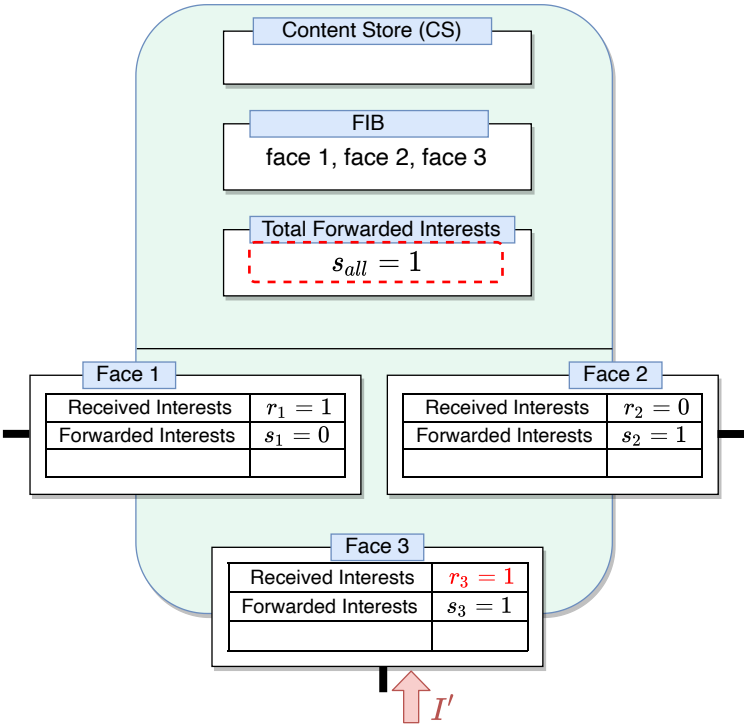


Figure 3.7 – Example scenario for Pessimistic Forwarding (after I)

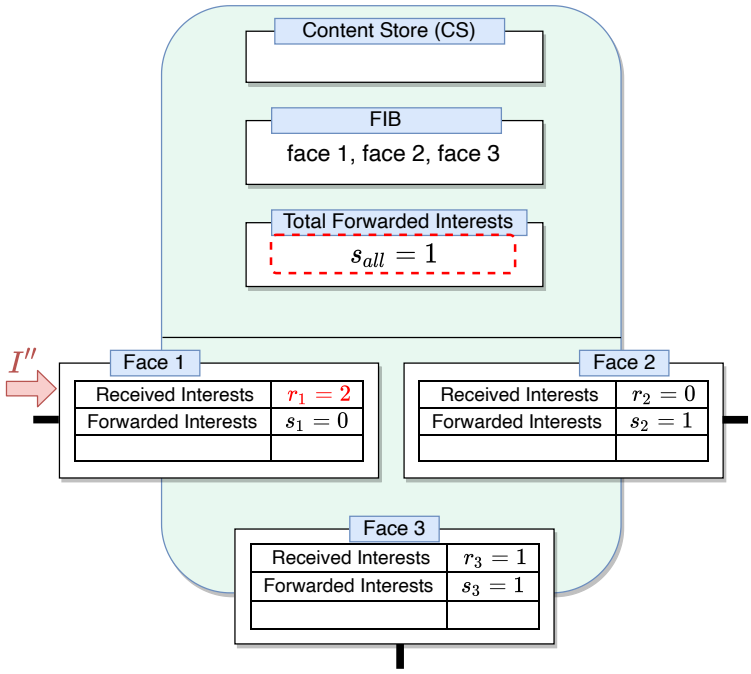


Figure 3.8 – Example scenario for Pessimistic Forwarding (after I')

Pessimistic Forwarding (PF)

Pessimistic Forwarding counts the number of times s_{all} that the node had to forward Interests. Forwarding is done when the number of Interest received at any face f becomes more than the number of times the node has forwarded Interests, *i.e.*, $r_f > s_{\text{all}}$. This is one interpretation of the proposition in [68].

There is a subtle ambiguity because [75, p.6] reads "For the sake of simplicity, we make the assumption that nodes follow a simple model in which any forwarded Interest brings an innovative segment before its expiration." However, if an Interest is forwarded on three other faces, does it count as one "forwarded Interest" or three? PF assumes one and OF assumes three. Now we consider that the node implements Pessimistic forwarding in the scenario in Figure 3.5.

- The node forwards the first Interest since $r_1 > s_{\text{all}}$. This happens because all the counters are initially set to zero, except the counter r_1, updated upon receiving the first Interest I.
- When the second Interest I' arrives, the node would not relay it as $r_3 = s_{all}$. The counter s_{all} remains unchanged. Figure 3.7 shows the state of the node at the arrival of the second Interest.
- The third Interest I'' would be forwarded since r_2 will be incremented, and consequently, the condition $r_1 > s_{\text{all}}$ will be satisfied ($r_1 = 2$ while $s_{all} = 1$). Figure 3.8 indicates the state of the node at the arrival of I''. After the Interest is forwarded, the counters s_2, s_3 and s_{all} will be updated.

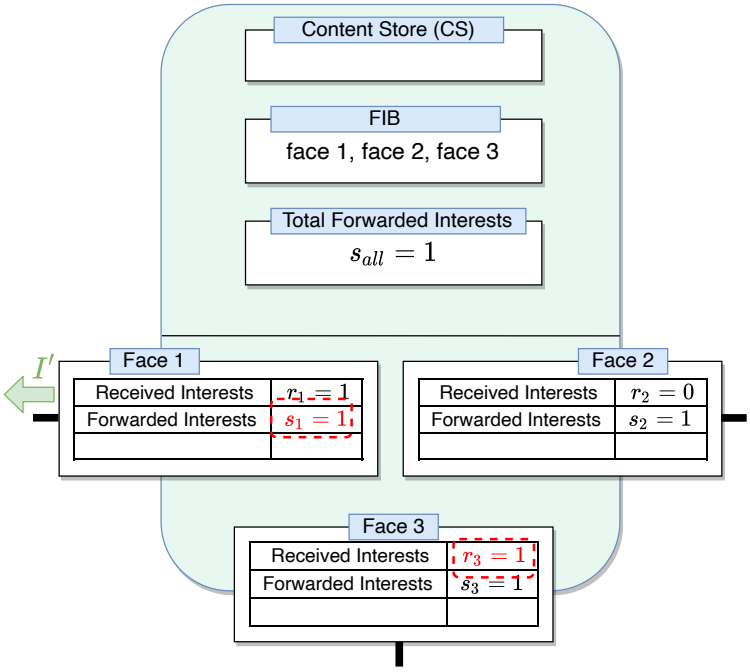


Figure 3.9 – Example scenario for Robust Forwarding

Robust Forwarding (RF)

The Robust Forwarding forwards Interest on each interface $\varphi \in \mathscr{F} \setminus \{f\}$ where less Interests have been forwarded than received on f, *i.e.*, $s_\varphi < r_f$. Now we consider that the node implements Robust forwarding in the scenario in Figure 3.5.

- The Initial Interest I is forwarded similarly as in OF and PF since the CS is empty.

- The second Interest I' that arrives at face 3 will be forwarded on face 1 since $s_1 < r_3$. Figure 3.9 presents the state of the node with Robust forwarding after the forwarding decision has been taken for the second Interest.

- Similarly, the third Interest I'' will be forwarded on faces 2 and 3, since $s_2 < r_1$ and $s_3 < r_1$ (r_1 is incremented on the third Interest's arrival).

With Robust Forwarding, all three Interests in the given scenario will be forwarded by the node. Notice if the Interests were distinct and duplicates were dropped or merged upon arrival, or if each client originates a single Interest; this mimics NDN semantics [34].

FIB Computation

The forwarding operation of a node also depends on how the FIB is populated. We describe several choices to fill the FIB.

- FIB $\mathscr{F}$ can include all faces of the node. This leads to a Flooding forwarding behavior. We denote it as FIB-F.

- The FIB can only include the faces that can lead to a source without looping back to the node. It is denoted as FIB-P.

- The FIB of all nodes could be configured to form a directed acyclic graph directed towards the sources. It is denoted FIB-DAG.

3.3.2 Logical Analysis of the Behavior of NetCodCCN Protocols

We assume clients and sources do not forward Interests and that a client sends each of its Interests on all its faces. For the analysis of the forwarding strategies, we only focus on the Interest forwarding achieved with the OF, PF, and RF; the data retrieval is not considered. Additionally, all the analysis is performed over networks with unit capacity links.

The first important forwarding design trade-off depends on the FIB. FIB-F is straightforward to implement since it includes all faces and forwards to all faces. FIB-P is often used instead of FIB-F [3], as it suppresses obviously useless Interests. However, it requires knowledge of paths to sources in order to avoid self-looping. FIB-DAG requires a potentially expensive FIB computation protocol (*e.g.,* NLSR [34]) and some coordination of the nodes to avoid loops. Thus FIB-DAG is only suited for applications in smaller networks, but it can reduce the use of network resources.

Looking in detail at the different FIB designs, we note two important properties:

[3]Not always clearly specified in all articles, but default behavior in some simulators such as ndnSIM [76] {https://ndnsim.net/2.1/doxygen/ndn-global-routing-helper_8cpp_source.html}is equivalent to FIB-P

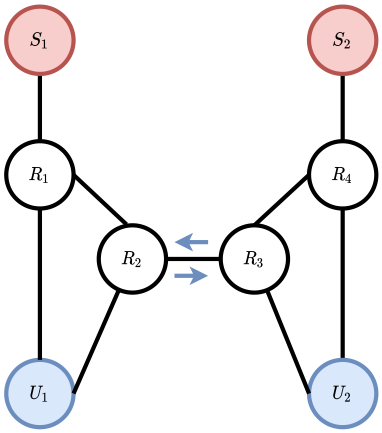


Figure 3.10 – Bidirectionality

1. **FIB-DAG is sometimes suboptimal**: This can be shown by the example scenario in Figure 3.10, where U_1 and U_2 are the clients requesting content stored at sources S_1 and S_2. We assume that all the links are unit capacity. The use of link R_2–R_3 in both directions is necessary for clients U_1 and U_2 to be able to receive simultaneous content from S_1 and S_2. However, the bi-directional use of the link R_2–R_3 ends up in creating a cycle that is incompatible with a DAG. Hence FIB-DAG is not always efficient and does not always allow maximum throughput.

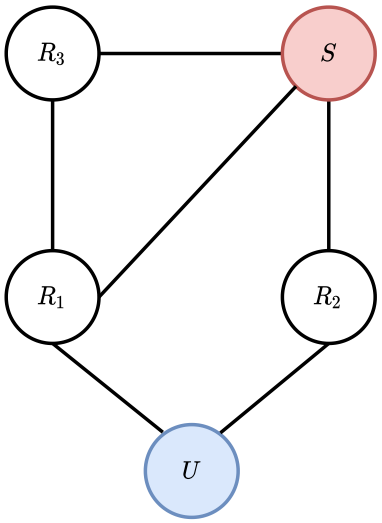


Figure 3.11 – Example network presenting overload

2. **FIB-P and FIB-F can generate extraneous traffic**: Pessimistic and robust forwarding allow content to be quickly delivered to clients, but we observed in our simulations (detailed in Chapter 4, Section 4.7) that Data packets could be exchanged long after the clients get all content, because of the generous replication of Interests. In Figure 3.11, we present a simple example with one client U sending k Interests to get back a k segment content. With parallel forwarding strategy RF and PF, the Interests will be forwarded to both the links U–R_1 and U–R_2 and then on over all the links on the next node to reach the source S. The client U will get all

content in $\frac{k}{2}$ time units as the content will be received from both links. However, node R_3 and R_2 that have also forwarded all the Interests will continue to receive content during k time units since they only have one link and only one packet can be received per unit time. The content that will be received after time $\frac{k}{2}$ by these nodes will be redundant for the client, but it continues to flow in the network. This superfluous traffic reduces the capacity available for other content or data traffic.

We now focus on the Interest forwarding algorithms: Optimistic Forwarding (OF), Pessimistic Forwarding (PF), and Robust Forwarding (RF). One fundamental and necessary property of one such algorithm is that the number of Interests that reach the source(s) is greater or equal to the number of Interests sent by the client(s). We denote this property *conservation* for scenarios where no sources actually reply with Data packets (or with near-infinite delay). The following analysis studies the preservation of the conservation property with the forwarding algorithms (OF, PF & RF).

OF Analysis With One Client

We first observe that OF has difficulties with conservation. This comes from the fact that a node of degree $d+1$, when receiving kd Interests from a face, will send k Interests on each forwarding face as shown in Figure 3.12. This is a form of bounded local flow conservation. With the OF strategy, the output rate (number of outgoing Interests) of one node is lower or equal to the input rate (number of incoming Interests), which is sufficient only if every Interest gets a linearly independent reply.

Consider the case in Figure 3.13, where we count the ratio of Interests flowing on each link compared to the number of Interests originated by the client U. Because of the previously noted property, the flow splits at node R_1. However, if the flow meets again as in R_4 in Figure 3.13, a proportion of the Interests will not be forwarded ($\frac{1}{4}$ in Figure 3.13). Hence, the total flow is not conserved as the source receives only $\frac{3}{4}th$ of the Interests originally sent by the client. Thus cross links prevent conservation under OF

With one client with one link (and any number of sources), conservation is possible with OF if and only if there are only parallel connections and no

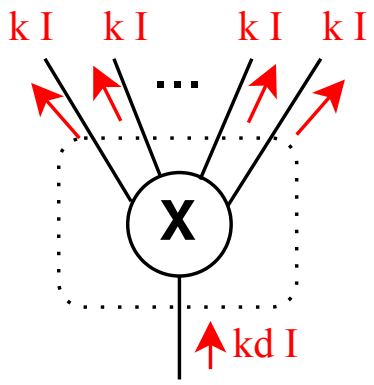


Figure 3.12 – OF forwarding

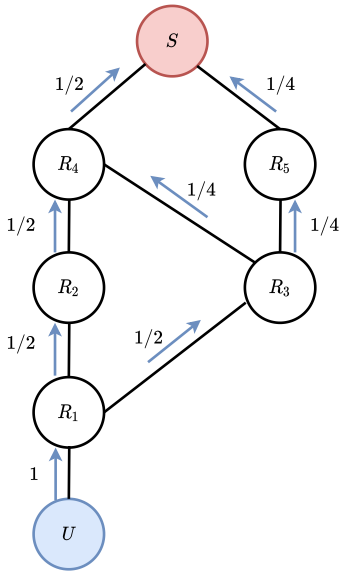


Figure 3.13 – OF with one client (links labelled with the ratio of Interest flow)

cross links, which implies a tree structure of the nodes (before the sources) as in Figure 3.14. Not even FIB-DAG guarantees this.

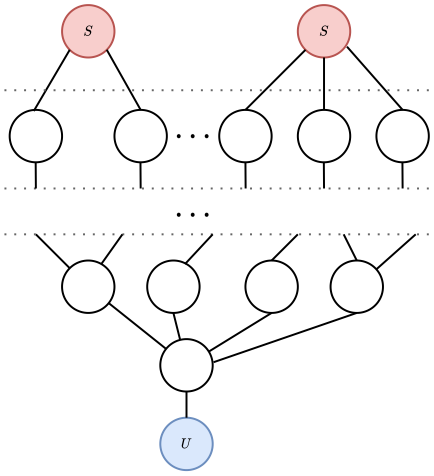


Figure 3.14 – Conservation in OF

OF Analysis With Several Clients

OF with several clients suffers from the same issues. However, in case of multiple clients requesting the same content, the Interest flow from a client might compensate flow losses for another client, as shown in the example of Figure 3.15, where conservation is observed this time. The same situation can also occur for a single client with several outgoing links.

As a result, we know that conservation is generally not verified for OF with several consumers, but the characterization of such cases appears more complex.

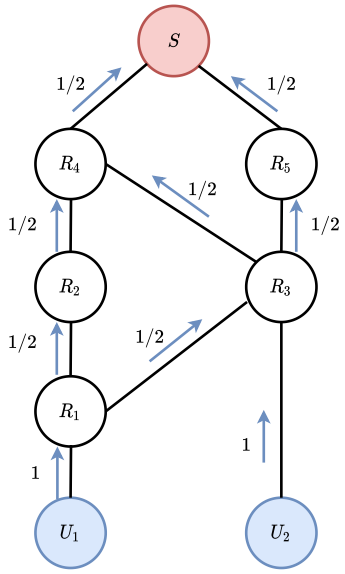


Figure 3.15 – OF case with several clients

RF Analysis

RF will always provide conservation. This comes from its property that any node that receives k Interests on one of its links will always forward k Interests on each of its other faces in the FIB (either the k Interests or some similar ones received previously). Figure 3.16 presents a example scenario to prove conservation for RF: it shows that if one has a graph G_i with that very property, one can expand it with any neighbor node to obtain a graph G_{i+1} with the same property, *e.g.*, k Interests coming in on the left, will still result into k Interests going out on the links of the new node N to the right (and vice-versa – not represented). Growing a graph iteratively (excluding clients and sources), one can prove recursively that any k Interests sent by a client will result in k Interests reaching every node of the network (FIB-F), or at least every node on the path to source(s) (FIB-P, FIB-DAG) and then every source.

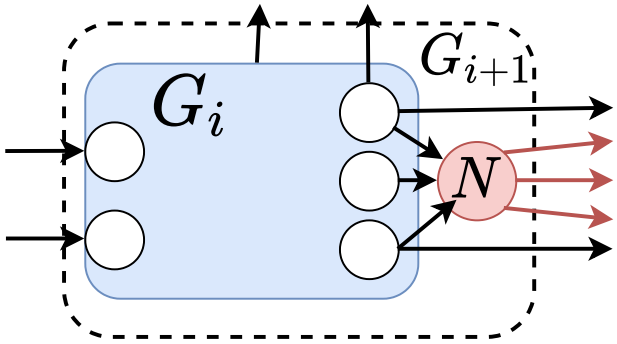


Figure 3.16 – RF Case

PF Analysis With Several Clients

We first observe that in some scenarios, with FIB-F, PF will not provide conservation. In Figure 3.17, if Interests are sent simultaneously by clients C_1 and C_2, the k^{th} Interest of C_1 is blocked by the k^{th} Interest of C_2 at node R_3 because it has already forwarded the latter.

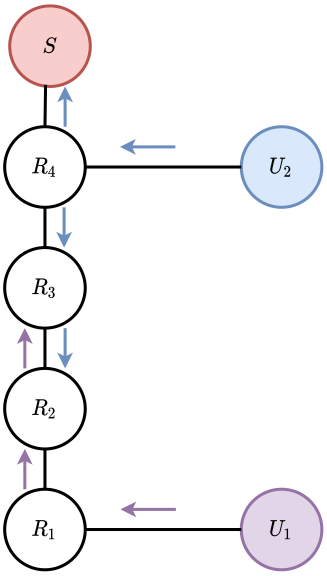


Figure 3.17 – PF case with several clients

PF Analysis with One Client

Under the assumption that client Interest have sufficiently large inter-arrival time[4], one can prove that with FIB-F, "if the (unique) client sends k Interests, then k Interests must reach every source". We denote this property $\mathcal{P}_k$.

We first introduce a concept derived from a notable property from PF: let $\ell > 0$ be an integer. Imagine a node u_i forwards its ℓ^{th} Interest (*i.e.,* its s_{all} reaches ℓ). By definition, this is only possible if it has just received the ℓ^{th} of $r_f = \ell$ Interests on some face f. This face is linked to a previous node u_{i-1}; we can similarly identify the previous hop u_{i-2} that caused u_{i-1} to forward its ℓ^{th} Interest, *etc.* and we establish a path, denoted ℓ-causality path, that starts from the client (no loop)With that definition, by induction on n, we prove the property $\mathcal{P}_n$: "n Interests sent by the client will always result in n Interests reaching each source". $\mathcal{P}_0$ is obvious.

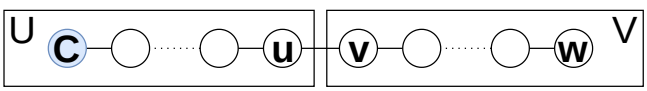


Figure 3.18 – PF Case

We then assume that $\mathcal{P}_n$ is true, and prove $\mathcal{P}_{n+1}$ by contradiction: we consider that $n+1$ Interests are sent by the client. Notice $\mathcal{P}_n$ can be applied to

[4] *i.e.,* when one arrives, the previous one has fully propagated

the first n Interests, hence at least n Interests will get forwarded by any node. Now denote U the set of nodes that forwarded exactly $n+1$ Interests, and V the set of nodes that forwarded n Interests (Figure 3.18).

Assume that V is not empty, and consider any node $w \in V$. Consider the n^{th} (and last) Interest forwarded by w, and the associated n-causality path of w. The n-causality path originates from the client c (in U), reaches $w \in V$, so it must cross the border between U and V: denote $u \in U$ and $v \in V$ such neighbor nodes on the n-causality path of w. By definition, the n^{th} forwarded Interest of u reaches v and is the n^{th} forwarded Interest by v.

Now we know by the definition that u will forward another $(n+1)^{th}$ Interest. This Interest cannot originate from v, therefore it will be received by v. It will be the $(n+1)^{th}$ received by v from u and v must forward another $(n+1)^{th}$ Interest. This contradicts the fact that v is in V. Hence the hypothesis "V is not empty" cannot be true, and the property $\mathcal{P}_{n+1}$ is verified for $n+1$, and by induction, for any n. ■

The analysis presented above gives a closer view of the forwarding techniques and their consequences. To conclude, some seemingly simple optimizations (such as OF, or PF with several clients, *etc.*) actually lead to losing the conservation property. Therefore, not every Interest will reach a source (or a cache). This is losing a vital design property (that, for instance, NDN maintains). Our solution will be to use robust versions of the algorithms such as RF; however, they also come with a cost (such as superfluous data traffic after the entire content has been retrieved by clients).

3.4 Conclusion

In this chapter, we have explored some design choices for network-coded ICN presented in the past and proposed novel enhancements. The first part of this chapter presented a solution for compressing the information in the Interests to ensure retrieval of linearly independent content.

In the second part of this chapter, we analyzed the properties of a family of protocols presented in the past that have been experimentally observed to reach network capacity. We focused mainly on the variants of their Interest forwarding algorithms and presented some essential properties of NetCodCCN concerning Interest Forwarding. We illustrated that different variants of forwarding algorithms lead to different trade-offs and properties: some are not necessarily easy to establish nor known as presented in the published work [68, 69, 73].

4

MICN: A Network Coding Protocol for ICN

4.1 Introduction

Chapter 2 illustrated the fact that integrating ICN and network coding yields throughput benefits in a multi-client and multi-source network. Nevertheless, to get these benefits, the clients in an ICN network need to ensure that each Interest sent in the network brings back innovative (linearly independent) content.

This chapter presents MICN, a protocol that integrates network coding over NDN. The proposed protocol enables fast retrieval of content over an NDN architecture in multi-client multi-source networks. The clients request coded content by sending indexed Interests that bring back innovative content.

We focus on the challenges in terms of naming the Interests requesting coded content in network-coded ICN scenarios. We present our naming approach by introducing an index in each Interest that imposes constraints on the coded segments that can serve as a reply to that Interest. This construction of Interests is called MILIC (Multiple Interests for Linearly Independent Content). The MILIC constraints ensure retrieval of linearly independent content with each Interest. We also propose MICN (MILIC-ICN), a protocol that uses MILIC construction to integrate network coding over NDN, as well as some optimizations to improve the performance of the protocol further.

The chapter is organized as follows: Section 4.2 presents the MICN protocol's basic architecture. Section 4.3 presents the desired properties of coded content to ensure linear independence with each received packet, introduces our solution MILIC and present the mathematical proofs of the properties. Sec-

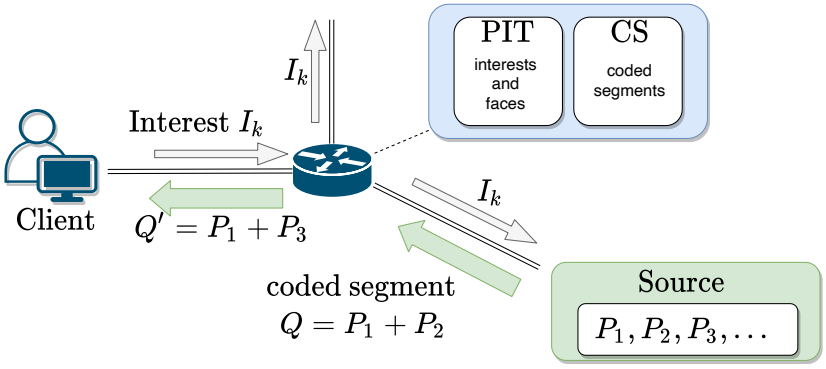


Figure 4.1 – Basic components of MICN (Source, Client, Interest, coded data, MICN node (PIT, FIB, CS)

tion 4.4 details the protocol specification of MICN. Finally, we present the simulation setup and the results of the MICN protocol in Section 4.7. We compare the performance with state of the art techniques.

4.2 Proposed Approach for Network-Coded NDN

In this section, we propose a protocol called MICN that supplements NDN with network coding. In MICN, the client nodes implicitly indicate the required coded segments by pipelining multiple distinct Interests. The distinct Interests allow parallel processing of these Interests and ensure that replies to the pipelined Interests are not redundant. We start by presenting an overview of the MICN architecture.

4.2.1 MICN Architecture

The architecture of MICN is based on NDN [34]. It incorporates common adaptations of network coding to NDN (as described in Section 2.6), including dividing content into generations and allowing in-network coding operations on the content from one generation. Figure 4.1 presents the basic elements of MICN. For the protocol messages: Interests are sent to retrieve coded segments, and replies to these Interests are coded segments (*i.e.,* linear combinations of the source segments) sent in Data packets. For the data structures: the nodes have a PIT, a CS, and a FIB. The data structures are modified to receive, process, and forward coded segments. Original or coded segments may be stored at the source and intermediate nodes. Each generation is retrieved independently of others.

One central design goal of MICN is to achieve high throughput using multiple available paths to the sources. To achieve this, MICN maximizes the propagation of Interests. The main MICN semantics are explained below.

- **Forwarding strategy:** MICN uses a *multicast forwarding strategy*: at each node, when possible, Interests are forwarded on all available faces.
- **Pipelining:** Interests are pipelined to allow parallel retrieval of content and hence increases throughput.

- **Interest processing:** Interests of different clients (similar Interests, different nonces) are not suppressed: this effectively allows the Interests of each client to reach the sources from multiple available paths.

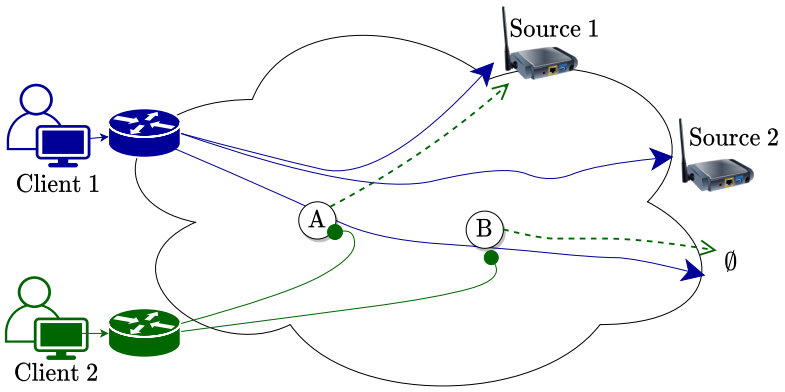


Figure 4.2 – Impact of NDN's Interest suppression: Interest from Client 2 suppressed due to earlier propagation of Client 1's interest

MICN, unlike NDN, chooses not to suppress the forwarding of similar Interests coming from different clients and treats each Interest with a different nonce as a new Interest. Figure 4.2 illustrates why MICN does not perform such merging of Interests. Interest I_k from client 1 in Figure 4.2 is propagated in the network first. The plain NDN semantics stops the propagation of similar Interests from client 2 since it expects the same content from the client 1 Interests. As a result, client 2 will be unable to take advantage of the multiple paths and will get content from only one path (from node A in Figure 4.2).

In order to achieve high throughput, one main challenge while pipelining and forwarding Interests to multiple paths is to ensure receiving linearly independent segments as replies. We impose constraints called MILIC (explained in Section 4.3) on the possible coded segments that can satisfy an Interest with index $i \in \{1, \ldots, n\}$: their encoding vector must follow constraints depending on i. MICN (through these constraints) ensures that coded segments sent as replies to Interests with different indexes are always linearly independent.

We adopt some additional features to complement these design choices. The most important additional feature is *Interest cancellation*. We introduce Interest cancellation as an option to cancel unnecessary Interests that may be lingering in the network after the retrieval of the content at the clients (see Section 4.5.2). *Just-in-time re-encoding* is adapted to prevent content from becoming non-innovative while being queued (Section 4.10). When there are multiple paths to a node, the fastest path is preferred. However, with pipelining, the contents tend to queue on the reverse fastest path. *Content redirection* allows all possible paths to be used (instead of just the fastest/ shortest path).

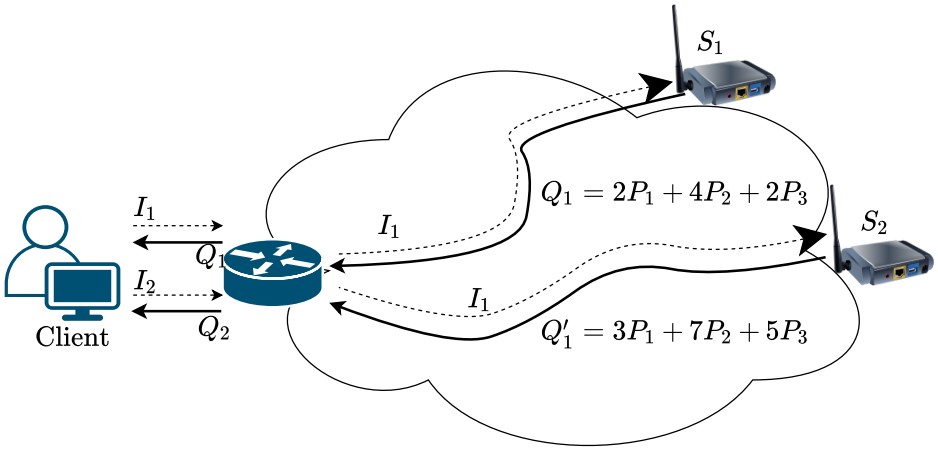


Figure 4.3 – MICN operation: Example scenario

4.2.2 Example of the Protocol Operation

Consider a single generation of size 3 with source segments P_1, P_2 and P_3 in $GF(11)$[1]. MILIC constrains any coded segment replying to the first Interest I_1: it should be a linear combination including P1, *e.g.,* $\alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$ with $\alpha_1 \neq 0$. A coded segment satisfying I_2 should not include P_1 and must include P_2, *e.g.,* $\beta_2 P_2 + \beta_3 P_3$ with $\beta_2 \neq 0$. Finally for I_3, the "coded" segment is just a scaling of P_3, *e.g.,* $\gamma_3 P_3$ with $\gamma_3 \neq 0$. Notice the constraints are realized with the subsets of encoding vectors $\mathcal{A}_1 = \{(v_1, v_2, v_3) \in \mathbb{F}_q^n \,|\, v_1 \neq 0\}$, $\mathcal{A}_2 = \{(0, v_2, v_3) \in \mathbb{F}_q^n \,|\, v_2 \neq 0\}$ and $\mathcal{A}_3 = \{(0, 0, v_3) \in \mathbb{F}_q^n \,|\, v_3 \neq 0\}$.

In the example of Figure 4.3 a client sends indexed Interest I_1 that is forwarded by the network to reach the two available sources. Each source sends a coded content $Q_1 = 2P_1 + 4P_2 + 2P_3$ and $Q'_1 = 3P_1 + 7P_2 + 5P_3$ satisfying the constraint for I_1 (*i.e.,* their encoding vectors are in subset $\mathcal{A}_1$), with encoding vectors $\begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \in \mathcal{A}_1$ and $\begin{pmatrix} 3 & 7 & 5 \end{pmatrix}$ respectively.

The intermediate node first receives the coded segment Q_1 and forwards it to the client as a response to the Interest I_1. On receiving Q_1 the client sends Interest I_2. Several sequences of events can happen that illustrate the main features of MICN:

- If the second content Q'_1 is received after I_2 then the intermediate node forwards I_2, since there is no content satisfying it that can be constructed from the CS. Later, more coded segment(s) will be received by the node, and I_2 will thus be satisfied. Note here that if the constraints were not imposed, the content Q_1 might as well serve as a reply for the second Interest as well as all later Interests. It is a critical to avoid this[2]. This is property 4.1 of MILIC (see Section 4.3).

- If the second content Q'_1 is received before I_2: the intermediate node will store the content Q'_1 along with Q_1 in its CS, since they belong to

[1] other Galois Field may be considered. With $GF(11)$ all coding coefficients are between 0 and 10 and all operation are modulo 11.

[2] Note that the method of NetCodCCN to avoid sending the same coded segment again, is to only reply to an Interest when the rank of its CS (here 1) is greater that the number of coded segments sent on the face (here also 1).

the same generation and are linearly independent. The CS may perform one step of Gaussian elimination, and keep them in row echelon form $R_1 = P_1 + 2P_2 + P_3 (= Q_1/2)$ and $R_2 = 1P_2 + 2P_3 (= Q_1' - 3Q_1/2)$. Now observe that when the Interest I_2 arrives, it can be satisfied at the intermediate node using the CS (this will illustrate property 4.3 of MILIC defined later in Section 4.3).

The example illustrates that although coded segments match different constraints and correspond to different subsets, all the coded segments of one generation are mixed together (in the decoding process, Gaussian elimination, and packet generation).

The constraints on the subsets and their properties are formally introduced in the following section.

4.3 Multiple Interests for Linearly Independent Content

4.3.1 Introduction and Motivation

The main goal in this work presenting network coding enabled NDN starts from the idea of consecutive transmission or pipelining where several Interests for content are sent in parallel, with the goal that each of them brings innovative coded segments. Additionally, to ensure that Interest/Data packets are not lost and the cache sizes are minimized.

The original content for network coding is partitioned into generations. With RLNC, segments from a given generation may be linearly combined within a source node or at any intermediate node in the network. The linear combinations are performed in some Galois field $\mathbb{F}_q$ to get *coded* segments. In what follows, $\mathbb{F}_q$ is a Galois field of q elements and $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. The coefficients in $\mathbb{F}_q$ of the linear combination form the *encoding vector* [56] of each coded segment.

For a generation of size n, we propose to use n distinct Interests. Interest $i \in \{1, \ldots, n\}$ can be satisfied by any coded segment whose encoding vector belongs to a predefined subset $\mathcal{A}_i$ of the set of all possible encoding vectors. In the following, these constraints will be such that the set of all non-zero encoding vectors is partitioned into n non-overlapping subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ satisfying some additional constraints.

4.3.2 Desired Properties of Subsets

To ensure that the answer to I_k is linearly independent to I_l, if $k \neq l$, the following properties of the subsets are sufficient conditions.

Property 4.1. For any $a_i \in \mathcal{A}_i, i = 1, \ldots, n$, the vectors $a_1, \ldots, a_n$ have to be linearly independent, *i.e.,* for any n-tuple of coefficients $(\alpha_1 \ldots \alpha_n) \in \mathbb{F}_q^n$, $\sum_{i=1}^{n} \alpha_i a_i = 0$ iff $\alpha_1 = \cdots = \alpha_n = 0$.

An additional condition can be imposed on subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ to benefit from the observation that when a node sends the same Interests over ℓ faces, ℓ

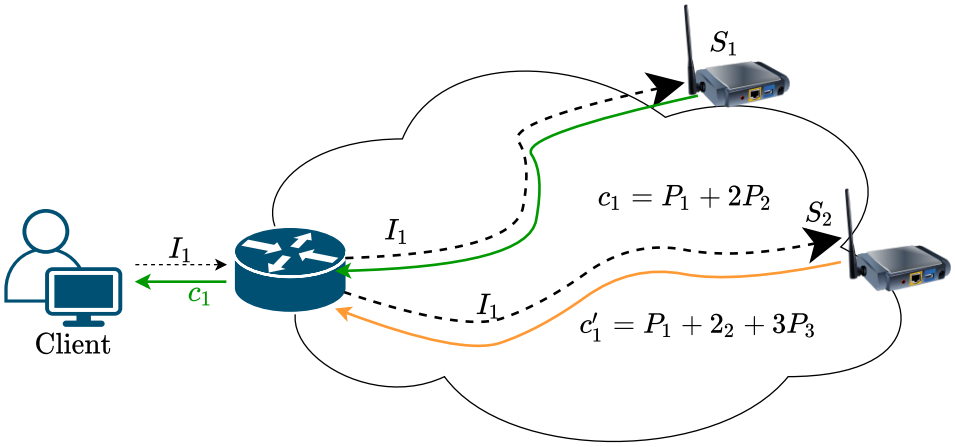


Figure 4.4 – Example explaining Properties 4.1 & 4.2

answers to these Interests will likely be received. Ideally, these replies should be linearly independent. This leads to a property of subsets that is not mandatory, but desirable to improve the efficiency of the proposed solution.

Property 4.2. Consider k distinct subsets $\mathcal{A}_{\pi(1)}, \ldots, \mathcal{A}_{\pi(k)}$ where π is a permutation of the integers 1 to k. Consider $\ell \geqslant 1$ vectors $a_\kappa^1, \ldots, a_\kappa^\ell$ chosen uniformly at random from each subset $\mathcal{A}_{\pi(\kappa)}$, $\kappa = 1, \ldots, k$ such that $\ell k \leqslant n$, then rank $\left(a_1^1, \ldots, a_k^\ell\right) = \ell k$ with high probability.

Finally, one may try to exploit the fact that segments are coded with possible re-encoding at intermediate nodes. Intermediate nodes may have received several segments with coding vectors belonging to the same subset. It may be of interest if these segments can be used to generate a coded segment with encoding vector belonging to another subset, to satisfy Interests for that subset. This translates into the following additional desirable property for the subsets.

Property 4.3. Consider the subset $\mathcal{A}_i$, $i = 1, \ldots, n-1$. For any pair (a_i^1, a_i^2) of linearly independent vectors belonging to $\mathcal{A}_i$, with high probability, there exist $\alpha_1 \in \mathbb{F}_q^*$ and $\alpha_2 \in \mathbb{F}_q^*$ such that $\alpha_1 a_i^1 + \alpha_2 a_i^2 \in \mathcal{A}_k$ with $k > i$.

Figure 4.4 presents a scenario that explains Properties 4.1 and 4.2. The client sends Interest I_1 that is duplicated by an intermediate node and sent on two different faces. The two Interests I_1 bring content coded with encoding vectors from the subset $\mathcal{A}_1$. With Property 4.3 there is a high probability that the two responses c_1 and c_1' are linearly independent and the node receiving these responses for I_1 has rank 2. The intermediate nodes can store the two responses in the CS since both are linearly independent. With the Property 4.3 the intermediate node can use these two responses to respond to an Interest I_2 from the client. These properties enable each MILIC Interest to bring informative content due to linear independence within the subsets along with linear independence between the subsets.

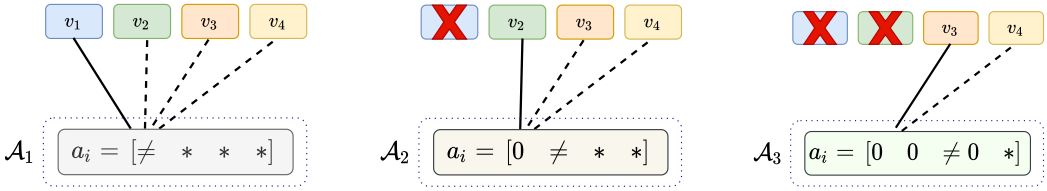


Figure 4.5 – MILIC subsets $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ construction for $\mathbb{F}_q^4$

4.3.3 MILIC Construction

In this section we propose a construction of the sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$, called MILIC, that satisfies the above properties. Consider

$$\mathcal{A}_i = \left\{(v_1, \ldots, v_n) \in \mathbb{F}_q^n \,|\, v_i \neq 0 \text{ and } \forall j < i, v_j = 0\right\}, \tag{4.1}$$

with $i = 1, \ldots, n$. With this construction the sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ form a partition of $\mathbb{F}_q^n \setminus \{(0, \ldots, 0)\}$.

In the remaining part of this section we present the proofs of how our presented construction leads to subsets that satisfy the desired properties presented in Section 4.3.2.

4.3.4 Proofs of MILIC Properties

Lemma 4.1. The cardinal number of $\mathcal{A}_k$ verifies $|\mathcal{A}_k| = (q-1)\, q^{n-k}$

Proof. Consider first $\mathcal{A}_1$: $\forall a_i \in \mathcal{A}_1$, one has $a_{i,1} \neq 0$. There are q^{n-1} vectors with leading zeros in $\mathbb{F}_q^n$ hence $|\mathcal{A}_1| = (q-1)\, q^{n-1}$. Then consider $\mathcal{A}_k$ with $k > 1$: $\forall a_i \in \mathcal{A}_k$, one has $a_{i,j} = 0$ for $j = 1, \ldots, k-1$ and $a_{i,k} \neq 0$. For $a_{i,k}$, we have $q-1$ possible choices. Then each $a_{i,j}$, $j = k+1, \ldots, n$ may take q possible values. Consequently $(a_{i,k+1}, \ldots, a_{i,n})$ may take q^{n-k} possible values and $|\mathcal{A}_k| = (q-1)\, q^{n-k}$. □

Thus, with the construction (4.1), the cardinality of the sets is decreasing with k. This implicitly imposes an ordering among these sets.

We now describe how Properties 4.1, 4.2, and 4.3 are satisfied by the sets defined by (4.1).

Property 4.1 is satisfied by construction: consider any $a_1 \in \mathcal{A}_1, \ldots, a_n \in \mathcal{A}_n$. The matrix with rows $a_1, \ldots, a_n$ is in row echelon form, and thus of full rank. The vectors $a_1, \ldots, a_n$ are thus linearly independent.

An example of the MILIC construction of the subsets $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ for a content with 4 source segments over $\mathbb{F}_q^4$ is presented in Figure 4.5. Set $\mathcal{A}_1$ according to the definition (4.1) consists of all the possible coding combinations of the source segments where the first coefficient must not be equal to zero and the rest of the coefficients can be chosen randomly from $\mathbb{F}_q$. The subset $\mathcal{A}_2$ consists of all the coding combinations such that the first segments is missing (*i.e.,* is equal to zero), the second must not be zero and the rest can be chosen randomly. The cardinal number of the subset $\mathcal{A}_2$ is smaller than the subset $\mathcal{A}_1$ and so on. Vectors chosen from each of the subsets, when stacked together create a matrix of the form presented in (4.2) and are linearly independent by

construction.

$$A_1 = \begin{bmatrix} \neq 0 & * & * & * \\ 0 & \neq 0 & * & * \\ 0 & 0 & \neq 0 & * \\ 0 & 0 & 0 & \neq 0 \end{bmatrix} \tag{4.2}$$

To prove Property 4.3 for $k > i$, consider an intermediate node that received two linearly independent segments $a_i^1 \in \mathcal{A}_i$ and $a_i^2 \in \mathcal{A}_i$. The $i-1$ first entries of a_i^1 and a_i^2 are zero, and their i-th entries $a_{i,i}^1$ and $a_{i,i}^2$ are non-zero. Then, as $\mathbb{F}_q^*$ is a group for multiplication, considering any $\alpha_1 \in \mathbb{F}_q^*$, there exists $\alpha_2 \in \mathbb{F}_q^*$ such that $\alpha_1 a_{i,i}^1 + \alpha_2 a_{i,i}^2 = 0$. Moreover, since a_i^1 and a_i^2 are linearly independent by assumption, one has $b = \alpha_1 a_i^1 + \alpha_2 a_i^2 \neq 0$. Let k be the smallest index such that $b_k \neq 0$. Necessarily $k > i$ and $b \in \mathcal{A}_k$.

Using Lemma 4.2 one deduces that Property 4.2 is satisfied for $\mathcal{A}_1, \ldots, \mathcal{A}_k$ as defined by (4.1) provided that $\ell \leqslant n - k + 1$, and evaluates the probability of having rank $\left(a_k^1, \ldots, a_k^\ell\right) = \ell$ and Lemma 4.3 illustrates Property 4.2 for the k first subsets $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

Lemma 4.2. Consider ℓ vectors $a_k^1, \ldots, a_k^\ell$ chosen uniformly at random from the set $\mathcal{A}_k$, $k \in \{1, \ldots, n\}$, and with $1 \leq \ell \leq n - k + 1$. The probability that $a_k^1, \ldots, a_k^\ell$ are linearly independent is

$$\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) = \prod_{j=1}^{\ell} \left(1 - \frac{q^{j-1} - 1}{(q-1)\, q^{n-k}}\right).$$

Proof. Consider first $\ell = 2$, and $a_k^1 \in \mathcal{A}_k$. The set of non-zero vectors collinear to a_k^1 and included in $\mathcal{A}_k$ is $\text{span}\left(a_k^1\right) \cap \mathcal{A}_k = \text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}$, whose size is $q - 1$. When choosing a second vector $a_k^2 \in \mathcal{A}_k$ uniformly at random, the probability that a_k^1 and a_k^2 are linearly dependent is equal to the probability that $a_k^2 \in \text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}$. Consequently, the probability that a_k^1 and a_k^2 are linearly independent is

$$\Pr\left(\text{rank}\left(a_k^1, a_k^2\right) = 2\right) = 1 - \frac{\left|\text{span}\left(a_k^1\right) \setminus \{(0, \ldots, 0)\}\right|}{|\mathcal{A}_k|} = 1 - \frac{1}{q^{n-k}}.$$

Assume now that the $j-1$ first vectors $a_k^1 \in \mathcal{A}_k, \ldots, a_k^{j-1} \in \mathcal{A}_k$ are linearly independent. The set of vectors that are linearly dependent with $a_k^1, \ldots, a_k^{j-1}$ and included in $\mathcal{A}_k$ is $\text{span}\left(a_k^1, \ldots, a_k^{j-1}\right) \cap \mathcal{A}_k = \text{span}\left(a_k^1, \ldots, a_k^{j-1}\right) \setminus \{(0, \ldots, 0)\}$. Its size is $q^{j-1} - 1$. Then, when choosing $a_k^j \in \mathcal{A}_k$ uniformly at random, the probability that $a_k^1, \ldots, a_k^j$ are linearly dependent is equal to the probability that $a_k^j \in \text{span}\left(a_k^1, \ldots, a_k^{j-1}\right) \setminus \{(0, \ldots, 0)\}$. Consequently

$$\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^j\right) = j \,|\, \text{rank}\left(a_k^1, \ldots, a_k^{j-1}\right) = j - 1\right) = 1 - \frac{q^{j-1} - 1}{(q-1)\, q^{n-k}}. \tag{4.3}$$

Then one has

$$\begin{aligned} &\Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell\right) \\ &\qquad = \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell, \text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell - 1\right) \\ &\qquad = \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^\ell\right) = \ell \,|\, \text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell - 1\right) \\ &\qquad\quad \Pr\left(\text{rank}\left(a_k^1, \ldots, a_k^{\ell-1}\right) = \ell - 1\right). \end{aligned} \tag{4.4}$$

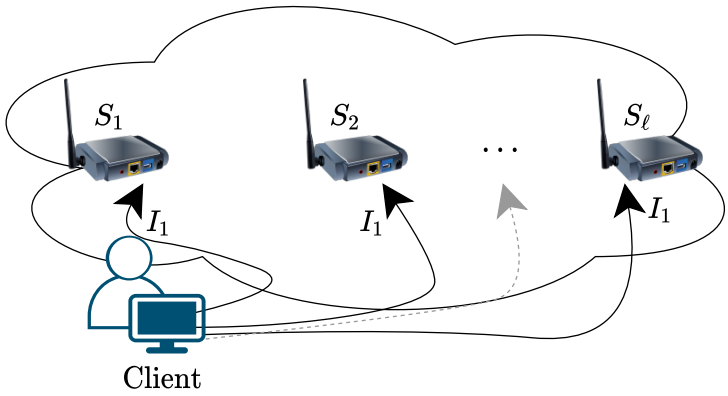


Figure 4.6 – Client requesting content for the subset $\mathcal{A}_k$ over multiple faces

Applying this recursively and using (4.3), one gets

$$\begin{aligned}
&\Pr\left(\operatorname{rank}\left(a_k^1,\ldots,a_k^\ell\right)=\ell\right)\\
&\qquad=\prod_{j=2}^{\ell}\Pr\left(\operatorname{rank}\left(a_k^1,\ldots,a_k^j\right)=j\,|\,\operatorname{rank}\left(a_k^1,\ldots,a_k^{j-1}\right)=j-1\right)\\
&\qquad\qquad\Pr\left(\operatorname{rank}\left(a_k^1\right)=1\right)\\
&\qquad=\prod_{j=1}^{\ell}\left(1-\frac{q^{j-1}-1}{(q-1)\,q^{n-k}}\right).
\end{aligned}$$

□

Example 4.1. Table 4.1 provides $\left[P_{\mathrm{F}}\left(\ell,1\right)\triangleq 1-\Pr\left(\operatorname{rank}\left(a_k^1,\ldots,a_k^\ell\right)=\ell\right)\right]$ for vectors of $n=10$ elements in $\mathbb{F}_{256}$ for different subsets $\mathcal{A}_k$ and different values of ℓ. One observes that choosing 5 vectors at random from any of the subsets $\mathcal{A}_k$, $k=1,\ldots,5$, results in a high probability of getting linearly independent vectors. Consequently, if a client sends 5 Interest packets for elements in $\mathcal{A}_k$ over different faces as illustrated in Figure 4.6, it is likely, provided that these Interests follow different paths leading to different sources or caches with independent content in the network, to get 5 linearly independent Data packets.

	$\ell=1$	$\ell=2$	$\ell=3$	$\ell=4$	$\ell=5$
$\mathcal{A}_1$	0	2.11×10^{-22}	5.46×10^{-20}	1.39×10^{-17}	3.58×10^{-15}
$\mathcal{A}_2$	0	5.46×10^{-20}	1.39×10^{-17}	3.58×10^{-15}	9.16×10^{-13}
$\mathcal{A}_3$	0	1.39×10^{-17}	3.58×10^{-15}	9.16×10^{-13}	2.34×10^{-10}
$\mathcal{A}_4$	0	3.58×10^{-15}	9.16×10^{-13}	2.34×10^{-10}	6.01×10^{-8}
$\mathcal{A}_5$	0	9.16×10^{-13}	2.34×10^{-10}	6.01×10^{-8}	1.53×10^{-5}

Table 4.1 – Probability of getting linearly dependent coded vectors of length $n=10$ chosen at random from $\mathcal{A}_k\subset\mathbb{F}_{256}^{10}$

Lemma 4.3. Consider $\ell\geqslant 1$ vectors $a_\kappa^1,\ldots,a_\kappa^\ell$ chosen uniformly at random from each subset $\mathcal{A}_\kappa$, $\kappa=1,\ldots,k$ such that $\ell k\leqslant n$. The probability that

$\left\{a_1^\lambda, \ldots, a_k^\lambda\right\}_{\lambda=1}^{\ell}$ are linearly independent is

$$\Pr\left(\operatorname{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{q^{j-1}}{q^{n-k}}\right).$$

Proof. According to Property 1, the vectors $a_1^1, \ldots, a_k^1$ are linearly independent. Consider the matrix A, whose first k rows are the vectors $a_1^1, \ldots, a_k^1$ and the $(\ell-1)\,k$ remaining rows are $\left\{a_1^\lambda, \ldots, a_\kappa^\lambda\right\}_{\lambda=2}^{\ell}$. The first k rows are used to perform Gaussian elimination on the $(\ell-1)\,k$ remaining rows to get a matrix A_1 of the form

$$A_1 = \begin{bmatrix} 1 & * & \cdots & & & * \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & & \\ & & & 1 & * \cdots * & \\ & & & 0 & & \\ \vdots & & & \vdots & B & \\ 0 & \cdots & & 0 & & \end{bmatrix}.$$

In A_1, B is a matrix of $(\ell-1)\,k$ rows and $n-k$ columns. Since all vectors chosen in the subsets $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, k$, have been selected uniformly at random, the $n-k$ last entries of each vector are independently and uniformly distributed. The i-th row of B results in a linear combination of $a_1^1, \ldots, a_k^1$ with one of the remaining vectors $a_\kappa^2, \ldots, a_\kappa^\ell$, $\kappa = 1, \ldots, k$. Consequently, the $n-k$ components of the i-th row of B are still independently and uniformly distributed. Since all $n-k$ last components of $a_\kappa^2, \ldots, a_\kappa^\ell$, $\kappa = 1, \ldots, k$ are independently and uniformly distributed; all components of the matrix B are independently and uniformly distributed.

The matrix A is of full row rank ℓk iff the matrix B is full row rank $(\ell-1)\,k$. The first row b_1 of B is non-zero with probability $1 - \frac{1}{q^{n-k}}$. The second row b_2 of B has components that are uniformly and independently distributed from the other entries of B and thus of b_1. The vectors (b_1, b_2) are linearly independent if b_2 does not belong to the space spanned by b_1. Since $\operatorname{span}(b_1)$ is of size q, one has

$$\Pr\left(\operatorname{rank}(b_1, b_2) = 2\right) = 1 - \frac{q}{q^{n-k}}.$$

Assume now that the $j-1$ first rows $b_1, \ldots, b_{j-1}$ of B are linearly independent. Under this assumption, the probability that b_j is such that the j first rows $b_1, \ldots, b_j$ of B are linearly independent is equal to the probability that b_j does not belong to the subspace of dimension q^{j-1} spanned by $b_1, \ldots, b_{j-1}$. Consequently,

$$\Pr\left(\operatorname{rank}(b_1, \ldots, b_j) = j \,|\, \operatorname{rank}(b_1, \ldots, b_{j-1}) = j-1\right) = 1 - \frac{q^{j-1}}{q^{n-k}}.$$

Then similarly as in 4.4, the probability that B is of full rank is given by

k	ℓ	$\mathbb{F}_2$	$\mathbb{F}_{256}$
50	2	0.71	0.0039
25	4	0.71	0.0039
33	3	0.42	1.53×10^{-5}
49	2	0.23	5.98×10^{-8}
48	2	0.06	9.13×10^{-13}
32	3	0.06	9.13×10^{-13}
24	4	0.06	9.13×10^{-13}
47	2	0.015	1.39×10^{-17}
45	2	0.00097	3.24×10^{-27}

Table 4.2 – Probability $P_{\mathrm{F}}(\ell, k) = 1 - \Pr\left(\mathrm{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right)$ of not getting ℓ linearly dependent vectors when choosing randomly ℓ vectors in each of the subsets $\mathcal{A}_1$ to $\mathcal{A}_k$

$$\Pr\left(\mathrm{rank}\left(B\right) = \left(\ell - 1\right) k\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{q^{j-1}}{q^{n-k}}\right) = \prod_{j=1}^{(\ell-1)k} \left(1 - \frac{1}{q^{n-k-j+1}}\right).$$
$$\approx 1 - \frac{1}{q^{n-lk+1}} \quad \text{when } q \text{ large}$$

Notice that the probability that B is full rank is close to one for appropriate parameter choices, as illustrated by Example 4.2. □

Example 4.2. Table 4.2 provides $P_{\mathrm{F}}(\ell, k) \triangleq 1 - \Pr\left(\mathrm{rank}\left(a_1^1, \ldots, a_k^\ell\right) = \ell k\right)$ for vectors of a generation of size n in $\mathbb{F}_q$ when choosing at random ℓ vectors from each of the $k = 50$ subsets. For $n = 100$, one observes that when a node receives 2 random packets from each $\mathcal{A}_\kappa$, $\kappa = 1, \ldots, 50$ subsets, provided that network coding is performed in $\mathbb{F}_{256}$, the probability of getting linearly independent packets is above 99.6%. The same result is obtained when 4 packets are obtained from each of the $k = 25$ first subsets. The constraints introduced by the subsets do not degrade the generation recovery performance significantly compared to plain network coding. This result is mainly obtained thanks to the fact that one considers packets received from the first (largest) subsets.

Remark 4.1. The size of the subsets $\mathcal{A}_k$ decreases when k increases, see Lemma 4.1. Thus, at first sight, considering the size of the subsets, given a set of random vectors (*e.g.*, cache of a node), there would be more possibilities to generate a coded segment with an encoding vector in the first subsets (larger) than in the last subsets (smaller). Nevertheless, due to pipelining behavior, this is not a problem. When there is a single path between a client and a source, Property 4.1 ensures that all contents are innovative. If however, ℓ distinct paths connect the client to one or more sources or different caches, the first Interest packet in the pipeline is replicated on the ℓ paths. This Interest should bring back ℓ linearly independent Data packets thanks to Property 4.2. Then again, thanks to Property 4.2, the k first pipelined Interest packets are likely to bring back $k\ell$ linearly independent Data packets, see Table 4.2. Consequently, when ℓ distinct paths connect the client to one or several sources, it is unlikely that

this client will need to send Interests for contents in the subsets $\mathcal{A}_k$ with k close to n. This opens the potential for an adjustment and optimization of the size of the pipeline.

4.4 MICN Protocol

This section complements the description of the MICN architecture introduced in Section 4.2. It focuses on the Interest and data processing using the MILIC construction presented in Section 4.3, to recover linearly independent content with each Interest in the context of NDN. We present in detail protocol specifications including the content fragmentation and naming, followed by the changes required in the basic NDN data structures, PIT, and CS in order to process Interests for MILIC coded content. Finally we present the Interest and content processing algorithms.

4.4.1 Content Segmentation and Naming

The original content C is partitioned into G smaller groups of segments, called *generations:* $C = [c_1, c_2, ..., c_G]$. Each generation c_g, $g = 1, \ldots, G$, contains n *equally-sized* segments $c_g = [c_{g,1}, c_{g,2}, ..., c_{g,n}]$. The network coding operations are restricted to segments that belong to the same generation and are assumed to be performed in $\mathbb{F}_q$.

A MILIC-compliant coded segment, with encoding vector in the subset $\mathcal{A}_i$, $i = 1, \ldots, n$, is defined as

$$\widetilde{c}_{g,i} = \sum_{j=i}^{n} \alpha_j c_{g,j}$$

with $\alpha_i \in \mathbb{F}_q^*$. The entries of $c_{g,j}$, $j = 1, \ldots, n$ and $\widetilde{c}_{g,i}$ are represented as elements of $\mathbb{F}_q$. Any coded segment $\widetilde{c}_{g,i}$ is identified by a prefix, a generation id g, a MILIC index i, and the encoding vector $\alpha = (0, \ldots, 0, \alpha_i, \ldots, \alpha_n) \in \mathbb{F}_q^n$ to indicate the weight of each source segment in $\widetilde{c}_{g,i}$. Consequently, we propose to identify $\widetilde{c}_{g,i}$ by the NDN name `<prefix>/MICN/<`g`>/<`i`>/<`$\alpha_i, \ldots, \alpha_n$`>` (`MICN` indicates that the content is network-coded). Other naming conventions are possible.

4.4.2 Requesting MILIC-Compliant Content

According to the naming convention of content segments, introduced above, the name carried by the Interest $I_{g,i}$ for a coded segment from C belonging to the generation g and with an encoding vector in $\mathcal{A}_i$ is `<prefix>/MICN/<`g`>/<`i`>`.

Contrary to other proposals integrating network coding to NDN/CCN, this Interest format allows the client nodes to pipeline multiple Interests for the same generation, provided that different indices i are specified in the names.

In practice, a client sends successive Interests for coded segments in a given generation g, starting from packets with encoding vectors in $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\rho$, where ρ is the pipeline size. Additional Interests are sent once the content starts flowing back. The pipeline size ρ limits the number of pending Interests from a client node at any time.

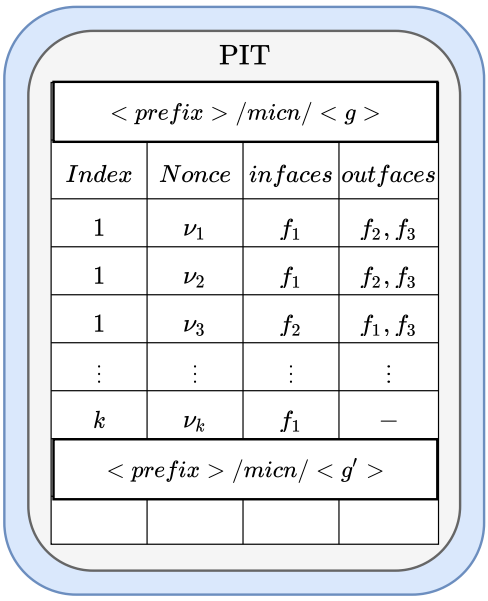


Figure 4.7 – MICN compliant PIT: the Interest with index k has a cache hit and is temporarily stored in the PIT until the queue of face f_1 is empty to send back the associated Data packet

Each Interest $I_{g,i}$ has an associated time-out. If linearly independent content in response to $I_{g,i}$ is not received before time-out, $I_{g,i}$ is sent again. Time-out may occur, *e.g.*, in case of loss of the Interest or Data packets.

4.4.3 MICN-Compliant PIT

Compared to the classical NDN PIT, a MICN-compliant PIT identifies Interests requesting coded segments with the same prefix and generation id as *related Interests*. PIT entries for related Interests are grouped in a sub-table (identified by the prefix and generation id g). Each entry itself includes the associated index i, nonce ν, as well as the *in* and *out* faces. The PIT entries are sorted by order of arrival.

Figure 4.7 illustrates a part of a MICN-compliant PIT at a given node with three faces f_1, f_2, and f_3. Three Interests have been received and forwarded. The two Interests associated with $\mathcal{A}_1$ are considered different since they have different nonces, which implies that different clients sent them. This is sufficient to implement the semantics that does not suppress similar Interests of different clients.

4.4.4 Just-in-Time Content Re-encoding/Replying

In plain NDN, whenever a node can satisfy an Interest, a copy of the requested content is sent immediately. In MICN, as in some other network-coded NDN protocols, nodes do not just forward a copy of the matching cached coded segment as an answer to the Interests. They linearly combine cached segments from the same generation to generate a new coded segment.

In MICN, the reply strategy is further modified, compared to plain NDN. A node waits until the output queue of a face is empty, before generating a coded

segment that satisfies a pending Interest on this face. This allows the node to use its latest cached contents when replying, hence sending more diverse content through the network. To achieve this, a one packet queue is considered at the faces, *i.e.,* only one Data packet is in transit on a face, the next is sent only when the packet in transit is completely delivered. The process to achieve this just-in-time re-encoding is further detailed in Sections 4.4.5 and 4.4.7.

4.4.5 Interest Processing

When a node receives an Interest $I_{g,i}$, it initially performs loop detection. If an Interest with the same nonce has already been received, $I_{g,i}$ is considered as looping Interest and deleted. Otherwise, the node can either reply using a content generated from its CS or further forward the Interest to the network.

CS Lookup

Like in the PIT, the *related contents* (*i.e.*, contents with the same prefix and generation id) are grouped in the CS. The CS can store related coded segments with network coding vectors in row echelon form. The *CS lookup* starts by identifying the related content matching the received Interest. A *cache hit* occurs if there exists a coded segment belonging to the subset requested by the Interest. This coded segment and all linear combinations of this segment with segments from subsets with a higher index can satisfy this Interest. Later, when generating a reply, a variant of RLNC can be used.

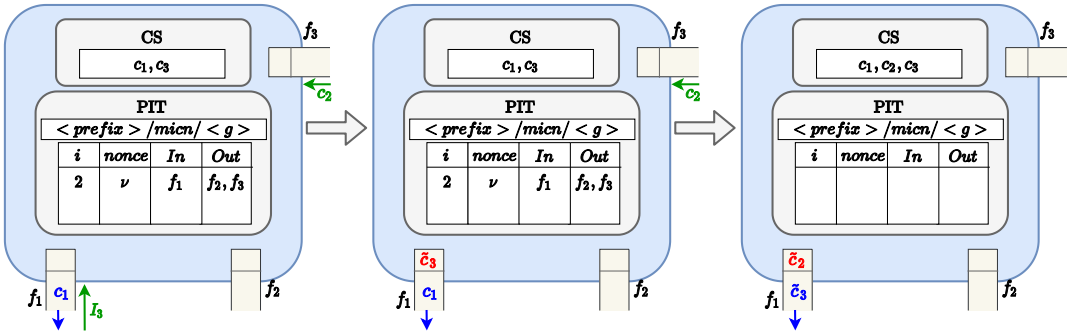
In case of a cache hit, the node schedules a reply for the Interest. The node first checks the outgoing queue of the face from where the Interest arrived. If the queue is empty, the content is immediately sent in a Data packet. Otherwise, a reply (linear combination) is generated only when the queue becomes empty. In our implementation, this scheduling is achieved by creating a *transient* PIT entry to store the incoming face, nonce, *etc.*, but without specifying an outgoing face, since the Interest does not require to be forwarded. See, for example, the Interest with index k in Figure 4.7.

Figure 4.8a illustrates a node that does not implement just-in-time re-encoding. When it has enough content in its CS to respond to the incoming Interest I_3, it immediately uses the related cached content to generate a response $\widetilde{c}_3$. However, the content remains in the queue until the content c_1 is transmitted. While the node in Figure 4.8b with just-in-time re-encoding waits until c_1 is transmitted, since it may receive more content and have a more diverse CS (since more content from the same generation is requested). So a transient PIT entry is generated that is replied to as soon as the queue becomes empty.

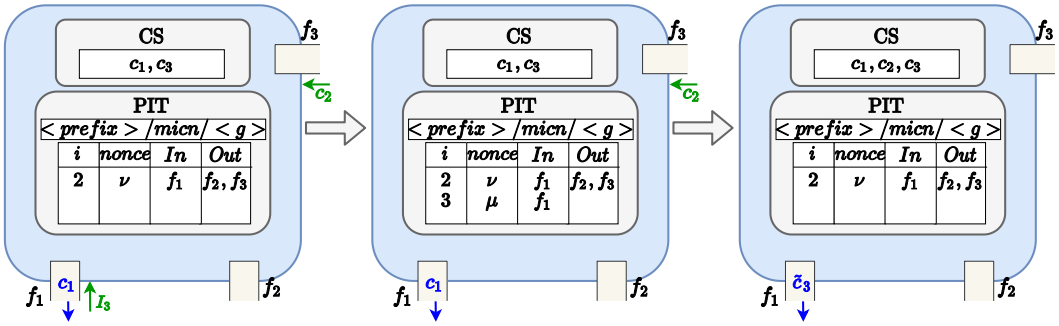
4.4.6 Interest Forwarding

In case of a cache miss, the node forwards the Interest to its next hop neighbors on available faces in the FIB (except the incoming face) and creates a PIT entry, which records the incoming and outgoing faces. Unlike classical NDN, different nonces result in different entries, see Figure 4.7.

Regarding the FIB management, multiple Interest forwarding strategies can be implemented depending on the subset of faces chosen to forward the content.



(a) Immediate re-encoding: $\widetilde{c}_3 = \alpha_1 c_1 + \alpha_3 c_3$ is put in the outgoing queue of face f_1 before the reception and processing of content packet c_2.



(b) Just-in-time re-encoding with MICN: $\widetilde{c}_3 = \alpha_1 c_1 + \alpha_2 c_2 + \alpha_3 c_3$ is put in the outgoing queue of face f_1 only once this queue is empty; this gives the opportunity to the later received c_2 on face f_2 to be included in $\widetilde{c}_3$.

Figure 4.8 – Two variants of cached content re-encoding

In MICN, to take advantage of multiple paths to the source(s) and to have the opportunity to receive multiple linearly independent segments, the FIB is filled with all faces that can lead to a source *without looping back* to the node. The multicast forwarding strategy is then used, in which Interests are forwarded on all faces in the FIB, as suggested in [34, Section 5.2.2].

4.4.7 Content Processing

When a coded segment arrives at a node, the node checks if the received segment and already cached related segments are linearly independent. If the encoding vector of the received segment belongs to a subset not present in the CS, it is immediately added to the CS (linear independence guaranteed by Property 4.1). If, however, the received coded segment has a network coding vector belonging to the a subset already in the CS, partial Gaussian elimination is performed to verify linear independence before adding it to the CS. The updated cache might then satisfy some additional Interests.

The node then uses its updated cache to reply to pending Interests. Whenever the queue of a face is empty, the node checks if any pending Interest on that face can be satisfied utilizing the current state of the cache. It answers the oldest PIT entry that may be satisfied and removes the entry.

4.5 Optimizations

In this section, we introduce some features that we adapt to complement our design choices compared to classical NDN. These adaptations help optimize the performance of MICN in an network-coded NDN scenario.

4.5.1 Content Redirection

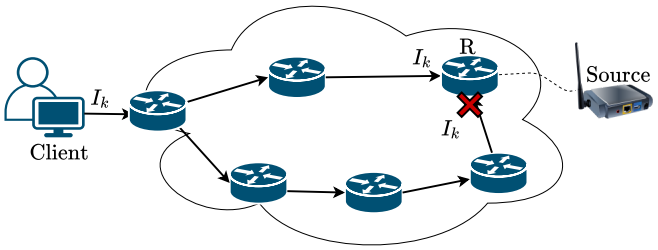


Figure 4.9 – Content redirection scenario

A node can receive an Interest on a second face while the same Interest (same nonce) is still pending at the first face, due to the Just-in-time content re-encoding of MICN, see Section 4.4.4. The duplicate Interest carrying the same nonce is considered looping and ignored. A common occurrence is shown in Figure 4.9 where one path is faster (*e.g.*, shorter) than the other. All Interests and, ultimately, content will queue on the faster path, while the slow path will not be used.

Content redirection attempts to utilize all available paths by exploiting the information brought by the looping Interests that there exists an alternate path to the client. Suppose the output queue associated with this alternate/second face is empty, and the node has matching content, then it can be immediately redirected to the client via this alternate face. This redirection is likely to improve the network utilization by benefiting from all paths leading to the client. In case of Figure 4.9, both links should be fully utilized.

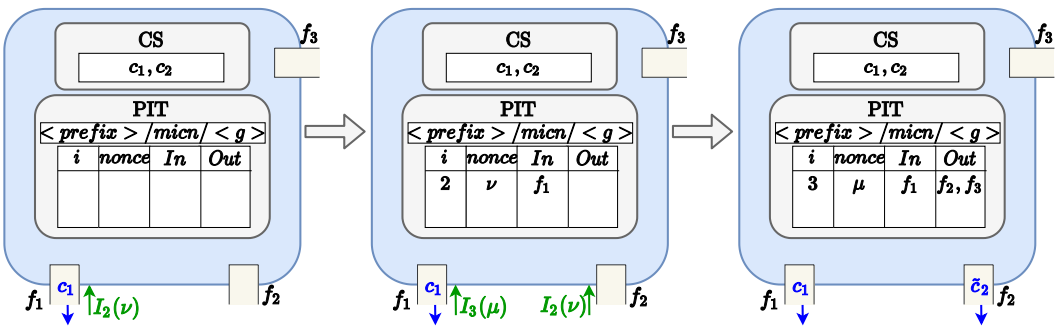


Figure 4.10 – Content redirection on face f_2: during the transmission of c_1, an Interest $I_2(\nu)$ for content associated to $\mathcal{A}_2$ has been received from face f_1 (left) and then from face f_2 (middle); since the outgoing queue of face f_1 is still occupied, $\tilde{c}_2$ is transmitted on face f_2 (right).

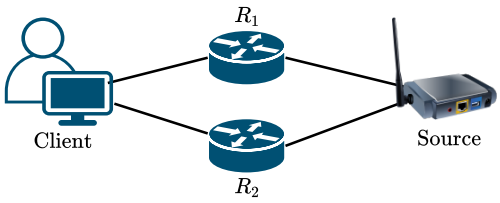


Figure 4.11 – Topology where Interest cancellation may be useful

Figure 4.10 depicts the state of a node that receives Interest I_3 with the same nonce ν from an alternate face f_2 with an empty queue. Since the node has enough content to generate a reply for the Interest, but the face f_1 is busy, the node redirects the content via the alternate face to immediately send the reply and possibly benefit from a second path to the client.

4.5.2 Interest Cancellation (MICN-IC)

We observe that content packets continues to flow in the network even after the client nodes have received enough content to decode a generation. This phenomenon is due to delay delay and connectivity discrepancies in different parts of the network which result in the extra Data packets flowing in the network. In order to reduce the traffic due to redundant contents, we introduce the concept of *Interest cancellation* (IC).

Figure 4.9 shows the simplest scenario justifying Interest cancellation. There are 2 paths between the client and the source, and Interests will be forwarded on both paths. The client will retrieve content at a rate twice faster than the nodes R_1 and R_2. There will still be pending Interests at the intermediate nodes when the transfer of content completes at the client. The content will still flow for these Interests; this content is redundant for the client. IC consists of having intermediate nodes deleting lingering Interests by using information carried by the new Interests on the content that is already retrieved at the client.

The optional *client identifier* and *state* fields are added in Interest packets to perform cancellation. The client identifier field can be any unique node identifier [3].The state field bears the information of subsets as defined by MILIC for which that client has already available content. Such content may have been directly obtained after Gaussian elimination involving several data segments. The state field can be represented by a bitmap indicating the available indices. Notice that client identifier and state fields introduce a small overhead on Interest packets.

When receiving an Interest with the state of a client, a node may ignore the pending Interests referencing indices i for which content is already available. Nevertheless, the nodes do not immediately delete them: instead, they are set to low priority for replies, contrary to other related Interests in the PIT, which have normal priority. Due to Properties 4.2 and 4.3, answering the low-priority Interests may still be useful: even network coding segments sent as replies for subsets for which content is already available at the client may bring new information with a high probability. Thus, a node first replies to Interests that are guaranteed to increase the rank of the client and then to Interests that are

[3]hash of the client node identifier or a randomly generated client nonce

likely to increase it.

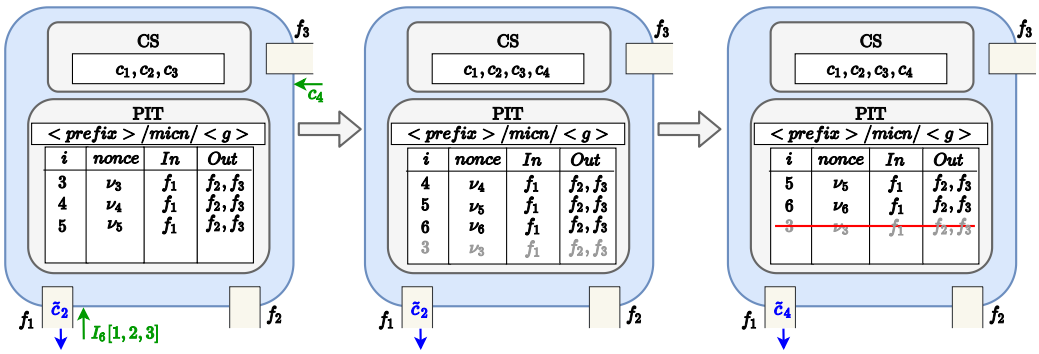


Figure 4.12 – Interest Cancellation: An Interest for packets associated to $\mathcal{A}_6$ is coming from face f_1, indicating that the source has already access to content associated to $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ (left); the pending Interest for content associated to $\mathcal{A}_3$ is first tagged with low priority (middle); this pending Interest is canceled as soon as an Interest associated to a subset of higher index (here $\mathcal{A}_4$) is replied to (right).

A reply to a low-priority Interest is sent only if the outgoing face is empty, and the node cannot generate content for Interests with a normal priority. Deletion of low-priority Interests occurs when the node has sent content for an Interest with a higher index than the low priority one to the client. This version of MICN with Interest Cancellation is referred to as MICN-IC.

Figure 4.12 illustrates the state of a node that receives an Interest for some content from $\mathcal{A}_6$. The Interest also carries the state of the requesting node, indicating that it has already access to contents from $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. Using this information the node sets the pending Interest for content from $\mathcal{A}_3$ to low priority (Interest in gray). This low-priority Interest is deleted once a content associated with $\mathcal{A}_4$ has been sent to the requesting client. Without Interest cancellation, the node would continue to process Interests and ensures that all pending Interests are responded to in FIFO order, based on the available content in the CS.

4.6 MICN Messages and Data-Structures

Table 4.12 compares MICN with classical NDN and NetCodCCN in terms of information stored in Interest and Data packets as well as in nodes. It gives an overview of the necessary information for the semantics of each protocol. For a given setting, it could also be the basis of a precise computation of each protocol's packet header size. In many cases, the network coding vector, whose size increases linearly with generation size, will dominate the overhead, for instance, for generation size 100 and $\mathbb{F}_{2^8}$[4].

[4]Remark that for MILIC, the network coding vector from index i has size $n - i + 1$

Interest Packet	
NDN	prefix, segment-id
NetCodCCN	prefix, gen-id
MICN	prefix, gen-id, MILIC-index
MICN-IC	prefix, gen-id, MILIC-index, client-id, bitmap (state)
Content Packet	
NDN	prefix, segment-id, content segment
NetCodCCN	prefix, gen-id, NC vector, coded segment
MICN	prefix, gen-id, MILIC-index, NC vector, coded segment
MICN-IC	prefix, gen-id, MILIC-index, NC vector, coded segment
PIT	
NDN	prefix, in-faces, out-faces, nonce-list
NetCodCCN	prefix, in-faces, out-faces, content counters [5]
MICN	prefix, in-face, out-faces, nonce[6]
MICN-IC	prefix, in-face, out-faces, nonce[6], priority

Table 4.3 – Information in Interest and Data packets as well as that stored in nodes (gen-id is generation identifier)

4.7 Evaluation

In this section we evaluate the performance of MICN over different topologies. The performance is evaluated in comparison to standard NDN and NetCodCCN [68]. We compare the protocols over a simple butterfly topology, which is a rather simple topology, that helps illustrate the properties and benefits of the protocol. The performance is also evaluated over a more elaborate topology close to the PlanetLab topology from NetCodCCN [68].

4.7.1 Simulation Setup

We implemented our simulator in Python. The simulator includes a generic packet network simulator (scheduler, link, packet transmission), on top of which we developed an implementation of the proposed MICN protocol. We also did lightweight reimplementations of NDN and NetCodCCN, capturing the main semantics of these protocols. This includes all the semantics described in Section 4.4.6 and the data structures, PIT, FIB, and CS, along with Interest forwarding (with suppression, management of similar Interests, multicast forwarding) in the spirit of [34]. For NetCodCCN, the main part is Interest processing, forwarding; we implemented the semantics as described in [68] (including Algorithm 1 and 2) [7].

At the link level, the parameters of our simulations are a propagation delay of 0.1 time units for each packet, a transmission time of 1 time unit for Data

[5] one counter per face for each prefix and generation id

[6] one entry per nonce

[7] Interest expiration was not implemented, since NetCodCCN assumes an expiration time large enough that any forwarded Interest will bring the requested segment before its expiration.

packets, and a small transmission delay for Interest packets ($1/(10 \times 2^{14}) \simeq 6 \times 10^{-6}$). A small amount of uniformly distributed transmission jitter (between 0 and $(1/10 \times 2^{21}) \simeq 3.8 \times 10^{-7}$) was also introduced.

In each topology, we consider the following scenario. Several clients request coded content, divided into generations of 100 segments each. We study the transmission of one generation. Each source stores a complete set of 100 segments. We assume that the intermediate nodes have enough cache space to store all segments of a generation. All the coding operations are performed in the finite field $\mathbb{F}_{2^8}$.

An Interest pipeline size $\rho = 10$ is considered, the FIB and Interest forwarding are as described in Section 4.4.6. At the client, each Interest packet has a time-out of 10 time units (*i.e.,* equivalent to the transmission delay of 10 Data packets, that is a bit longer than the longer round-trip delay). If a client does not receive innovative content for an Interest after this time interval, and the content has not yet been decoded from other data, it will resend the Interest.

For the results, the goal was to focus on the throughput of content. The performance is evaluated in terms of download time, *i.e.*, the time it takes for a client to download and decode a generation. Since we ultimately focus on throughput, all delays come from content transmission (assuming a fixed Data packet size). All the header overheads are ignored (see Table 4.3 for the overhead comparison). Packet processing delays (coding/recoding delays), buffer limits are neglected. An upper bound on the throughput (content/time unit) received by a client is given by the maximum flow of the graph from the sources to the client. From this max-flow, one can derive a lower bound on the download time. In similar settings, it had been proven that network coding could approach the max-flow bound [77], hence it represents a meaningful benchmark. Another metric of Interest is the total number of Data packets exchanged in the network until all clients have retrieved the generation with no Interest or Data packet present in the network anymore. MICN and NetCodCCN use a multicast Interest forwarding strategy; for fairness of comparison, a multicast strategy is also considered in NDN.

4.7.2 Results with the Butterfly Topology

We first analyze the behavior of MICN on a simple butterfly topology with two sources S_1 and S_2, and two clients U_1 and U_2, connected through a set of intermediate caching routers as represented in Figure 4.13. The butterfly topology is simple, however it evidences some of the properties of MICN.

The performance of the butterfly topology mainly depends on how the bottleneck link ($R_3 \leftrightarrow R_4$) is used. With classical NDN, the two clients U_1 and U_2 should request precisely the same segments on the middle link to improve performance. Nevertheless, the clients would require topology knowledge and coordination to do so. However, with network coding, client coordination or accurate topology knowledge is not required and the clients can simultaneously send their Interests to all their available faces. With network coding the clients can benefit from each others Interests.

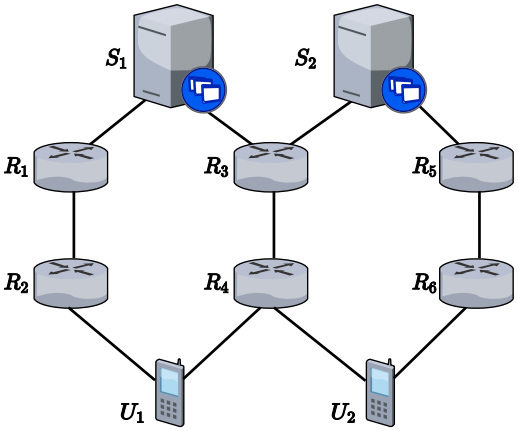
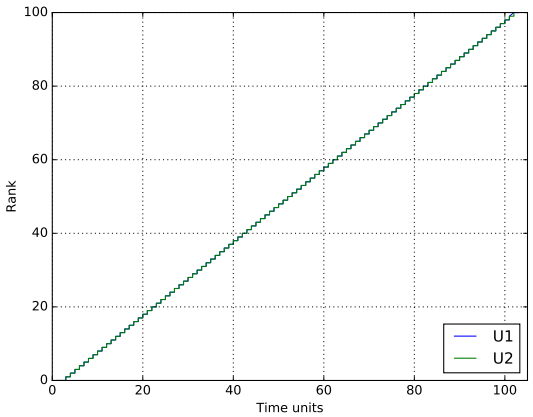


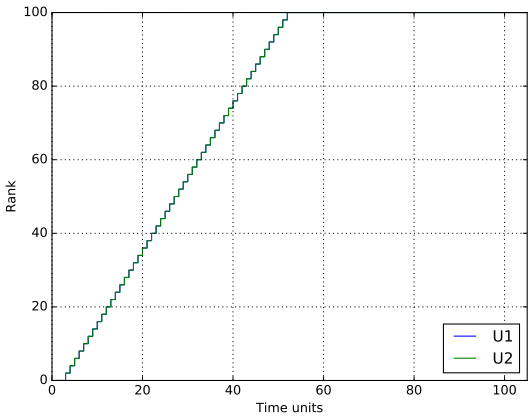
Figure 4.13 – Butterfly topology

Content Retrieval Time

Figure 4.14 shows the rank evolution of the client nodes over time for MICN, MICN-IC, NetCodCCN, and NDN. MICN, MICN-IC, and NetCodCCN retrieve content at the max-flow rate at each client, *i.e.*, each Data packet received at the client is innovative.



(a) NDN



(b) MICN, MICN-IC and NetCodCCN (all perform similarly)

Figure 4.14 – Butterfly topology: rank evolution as a function of time

After some initial delay, due to propagation, the clients receive 2 linearly independent Data packets every time unit, as shown in Figure 4.14b. The results presented in Figure 4.14b are for MICN, but the results are identical for NetCodCCN and MICN-IC. The similar download time for MICN, MICN-IC and NetCodCCN is confirmed by the final content retrieval time in Figure 4.15.

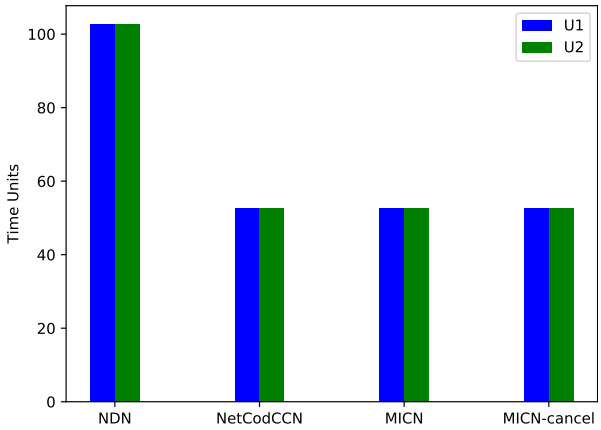


Figure 4.15 – Butterfly topology: content retrieval time

Data Traffic in the Network

The performance in terms of reaching maximum throughput is the same for all the network coding protocols, nevertheless, there are significant differences in the volume of data traffic that each protocol generates. Figure 4.16 presents the cumulative data traffic generated in the network with each protocol. Note that after NDN, NetCodCCN generates the most data traffic. MICN has a slightly reduced amount of data traffic, and MICN-IC has the least amount of traffic, which mostly accounts for innovative traffic.

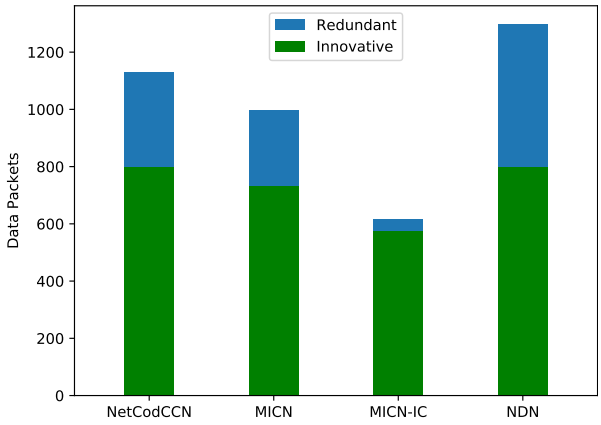
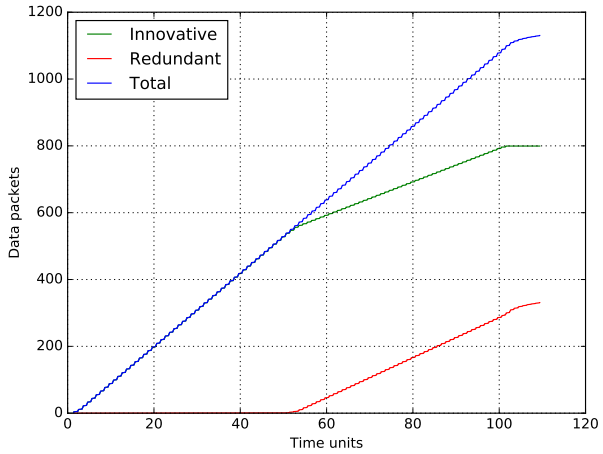


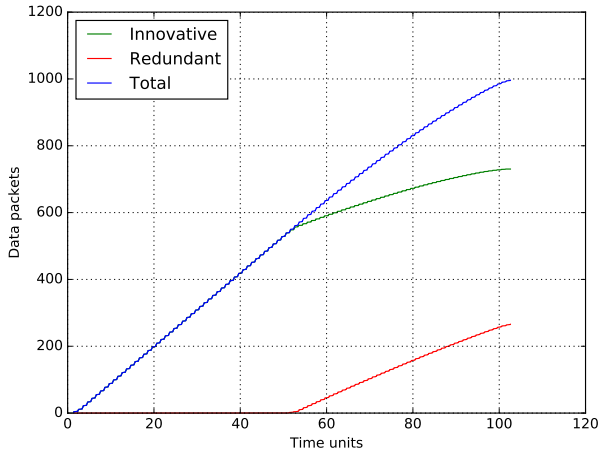
Figure 4.16 – Butterfly topology: cumulative data traffic

To understand the data traffic flowing in the network, Figure 4.17 depicts the evolution over time of the cumulative number of Data packets transmitted on all the links of the network, counted from time $t = 0$. The curves end when transmission of Data packets stops for the requested generation. In the beginning, there is only innovative traffic in the network, *i.e.*, all Data packets circulating in the network are innovative; for the intermediate nodes as well as the client node receiving them. Note that towards the end, when the clients have received the entire generation (around 52 time units), there is still data traffic flowing in the network. Since not all the network nodes have the same min-cut, they receive content at different rates. The intermediate nodes with a lower min-cut than the clients (see Figure 4.14) continue to get responses for

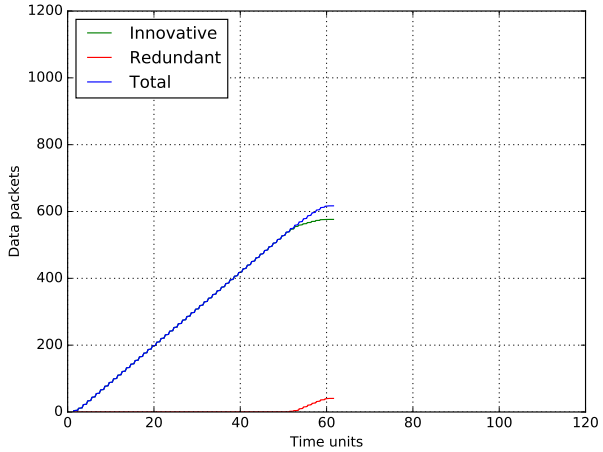
Interests that they have forwarded in the past. The content that arrives is still innovative for them. The intermediate nodes also continue to forward content to satisfy Interests in their PIT (no longer innovative for the clients; hence the redundant traffic curve starts to grow).



(a) NetCodCCN



(b) MICN



(c) MICN-IC

Figure 4.17 – Butterfly topology: evolution of cumulative data traffic as a function of time

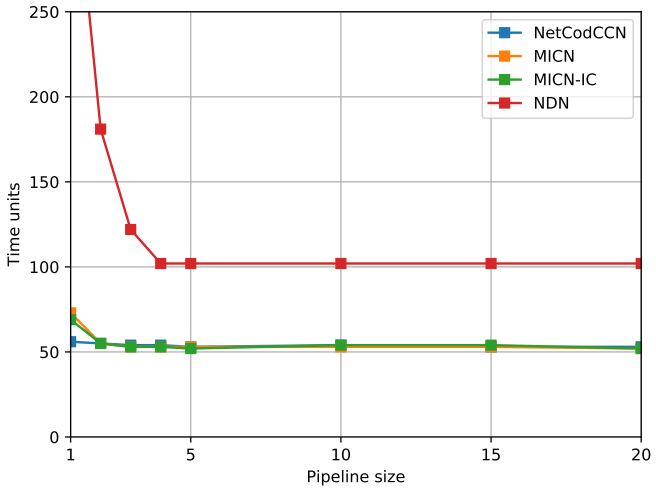


Figure 4.18 – Butterfly topology: impact of pipeline size (without losses) on download time

MICN-IC deletes Interests tagged with low priority, which are pending for a client that has access to content for those Interests. Canceling such Interests reduces the redundant data traffic, at the price of some signaling overhead. Precisely, in butterfly topology (Figure 4.13), 10 transmissions of Data packets over various links are necessary for delivering 2 Data packets to the clients U_1 and U_2, *i.e.*, 5 transmissions per packet. For a generation of size 100, a minimum of 500 transmissions are required for both clients to receive the entire generation. Figure 4.17 shows that with MICN-IC, a slightly larger amount of transmissions than the minimum are required. NetCodCCN achieves similar throughput, but Interests are not canceled, and several Data packets are redundant, leading to increased traffic.

Pipelining Effect

The effect of sending consecutive Interests by clients is analyzed in Figure 4.18. To have a continuous flow of content in the butterfly topology (in the absence of losses), the clients need to have at least two pending Interests at any time (because there are two paths) and usually even more because of the propagation delays. In the case of plain NDN with multicast strategy, the link $R_3 \leftrightarrow R_4$ becomes a bottleneck due to the absence of coordination among the clients. Even when the pipeline size increases, the performance cannot reach the one obtained with network coding. MICN, MICN-IC, and NetCodCCN, however, with a sufficient pipeline size (here as small as $\rho = 5$), can reach the maximum capacity.

Performance with Packet Losses

Next, we evaluate the performance of MICN in case of packet losses. Figure 4.19, depicts the effect of losses on the performance of the protocols. We consider transmission losses modeled with a fixed loss probability for both In-

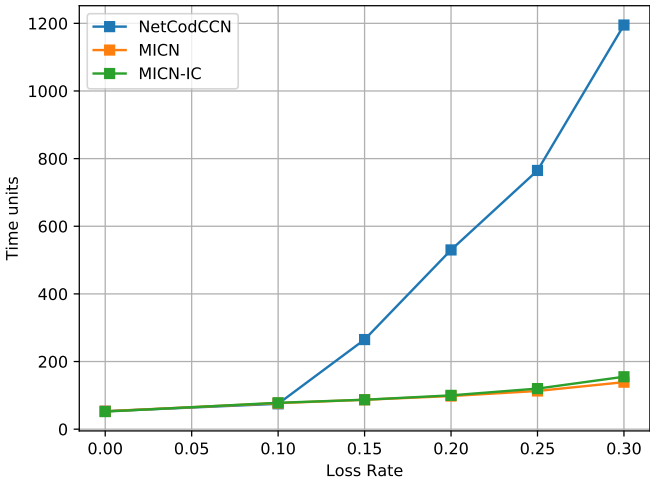


Figure 4.19 – Butterfly topology: impact of packet loss rate on download time

terest and Data packets[8]. MICN and MICN-IC appear to have much better performance compared to NetCodCCN. MICN has the advantage of precisely identifying which Interest (pointing to a subset $\mathcal{A}_i$) has timed out (no matching content received). In NetCodCCN, when a Data packet is lost downstream, the node will, nevertheless consider the Interest as satisfied (*i.e.*, update its content counters). An Interest retransmitted due to time-out is considered a new Interest, and the node will typically forward it. In MICN, the retransmitted Interest will be immediately satisfied by cache of the node.

4.7.3 Results with the PlanetLab Topology

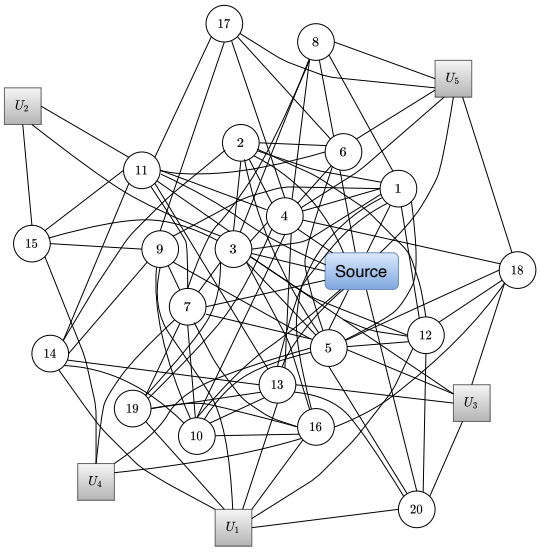


Figure 4.20 – PlanetLab Topology

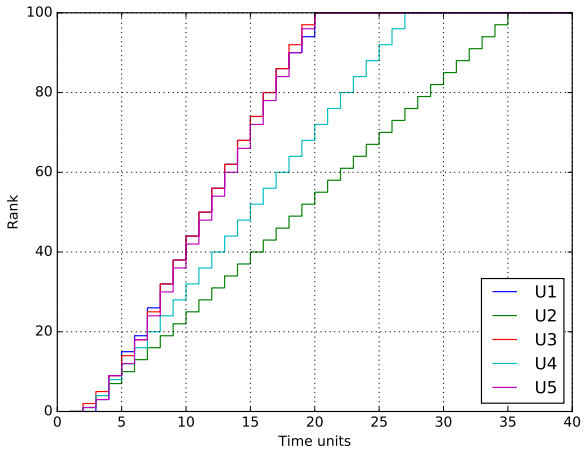
The behavior of MICN is then analyzed considering the PlanetLab topology (see Figure 4.20) adapted from [68][9], with one source and five client nodes

[8]Notice that NetCodCCN simulations in [68] consider only segment (Data packets) losses. However, here both Interest and Data packets are prone to losses.

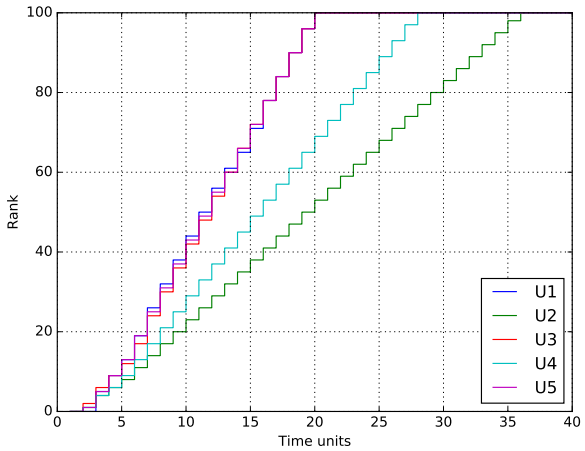
[9]originally adapted from the PlanetLab project [78]

connected through a set of 20 intermediate caching routers. All the links in the PlanetLab topology are equal capacity links.

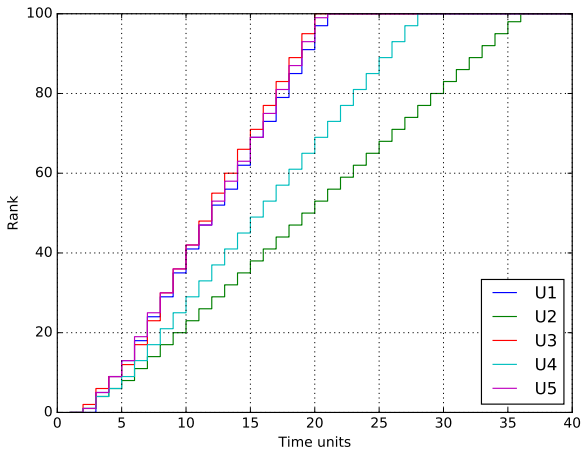
Content Retrieval Time



(a) NetCodCCN



(b) MICN



(c) MICN-IC

Figure 4.21 – PlanetLab topology: rank evolution as a function of time

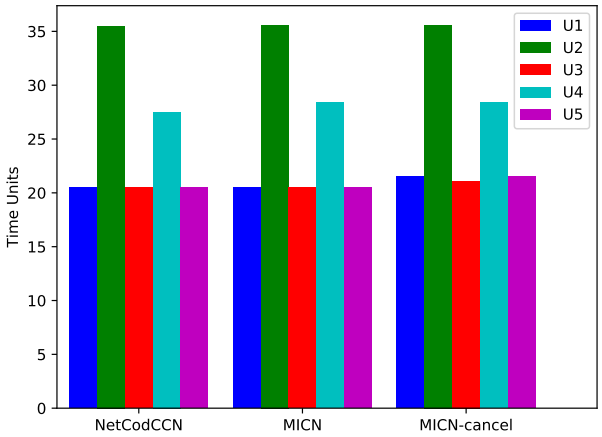


Figure 4.22 – PlanetLab topology: content retrieval time

Figure 4.21 shows the rank evolution of the client nodes over time for MICN, MICN-IC, and NetCodCCN. Note that the clients have different min-cut and the difference can be recognized in the graphs in Figure 4.21.

As seen in Figure 4.22, with MICN, MICN-IC, and NetCodCCN, clients receive enough content to decode a generation at a rate above 95% of the maximum rate (provided by the min-cut between the source and the clients), as observed for the butterfly topology.

Data Traffic in the Network

The difference between the data traffic generated by each network is represented in Figure 4.23. MICN has a better performance in terms of total traffic compared to NetCodCCN.

Figure 4.24 illustrates the cumulative number of Data packets transmitted on the network as a function of time. NetCodCCN generates the most data traffic (also for a longer duration). MICN is able to reduce the cumulative traffic by a considerable amount since the content does not have to be flooded on all the links as for NetCodCCN. MICN-IC performs best in terms of traffic, with respectively 2.16 and 3.42 times fewer transmitted Data packets compared to MICN and NetCodCCN.

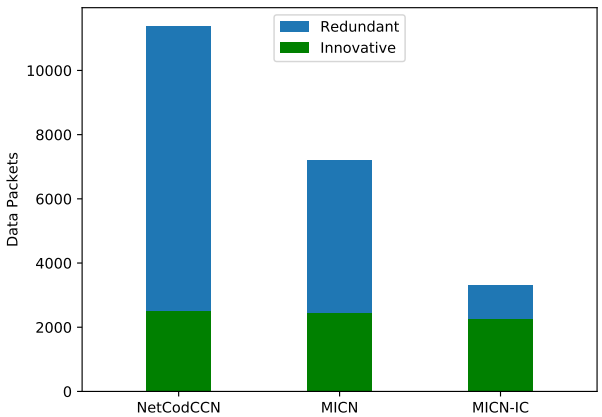
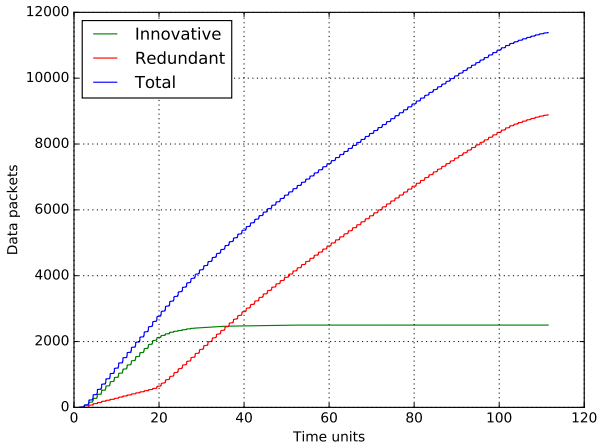
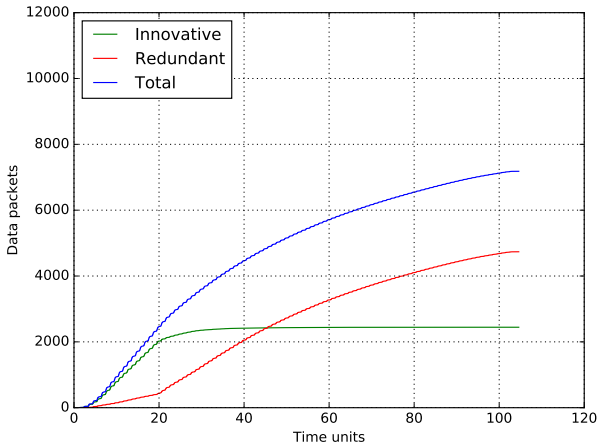


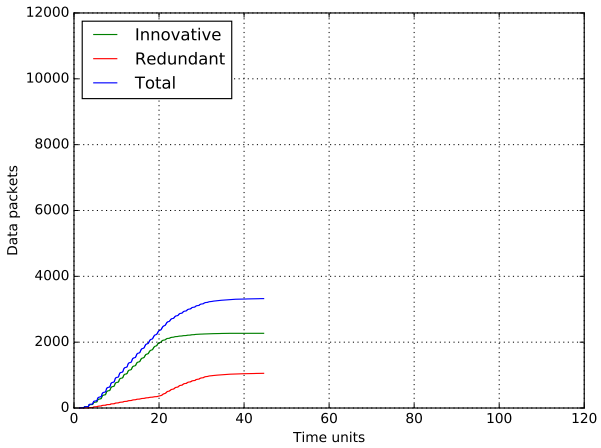
Figure 4.23 – PlanetLab topology: cumulative data traffic



(a) NetCodCCN



(b) MICN



(c) MICN-IC

Figure 4.24 – PlanetLab topology: evolution of cumulative data traffic as a function of time

In the PlanetLab topology, with the considered scenario, the amount of non-innovative packets dominates: about 80% of the content traffic with NetCodCCN is redundant (non-innovative). Note that not all the innovative traffic that appears in the cumulative traffic might not be useful for the clients. The graphs in Figures 4.23 and 4.24 represent the innovative traffic at all the nodes in the network and since intermediate nodes of the network are unable to detect

when a client has received all packets required to decode a generation. Consequently a Data packet that is innovative for intermediate node is considered innovative.

Pipelining Effect

In the PlanetLab topology, the pipeline size impacts the performance only when it is too small, as shown in Figure 4.25. Increasing the pipeline size above 2 (MICN), 3 (MICN-IC), and 5 (NetCodCCN) does not bring additional benefit.

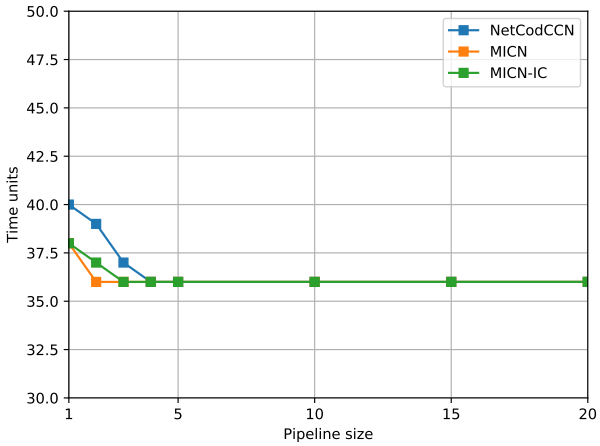


Figure 4.25 – PlanetLab topology: impact of pipeline size (without losses)

Performance with Packet Losses

Figure 4.26 shows the effect of packet losses. The download time with MICN and MICN-IC increases linearly with the loss rate, compared to NetCodCCN, which increases faster when the loss rate is above 10%. In the PlanetLab topology, compared to the butterfly topology, MICN, MICN-IC, and NetCodCCN are all more robust to packet losses due to the more significant amount of redundant content traffic in the network, which helps to compensate for the losses.

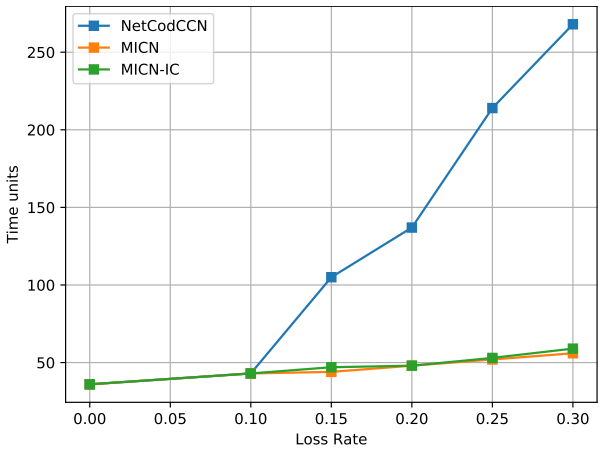


Figure 4.26 – PlanetLab topology: impact of packet loss rate

4.8 Conclusions

In this chapter, we studied and discussed the objectives and challenges of integrating network coding with NDN. We proposed a novel way of integrating network coding and Interest-based ICN. The proposed MICN protocol is built around the MILIC construction, that allows the clients to request content with encoding vectors that belong to predefined subsets by adding an index in the Interest, indicating the subset. This Interest naming allows the nodes to send multiple Interests in parallel and ensures that linearly independent content is sent as reply. In the considered scenarios, the clients download content close to their maximum capacity.

We observed that integrating network coding with NDN brought throughput benefits in multi-client multi-source scenarios where clients request a large content. We also observed that protocols that achieve higher throughput with this integration also incur higher costs in-terms of total data traffic generated in the network. We provided a solution for our protocol to reduce this higher traffic cost by introducing Interest cancellation. MILIC-IC limits the redundant data traffic considerably by adding some additional information in the Interests. This reduces the network load and leaves free network resources to fetch content from consecutive generations. In the next chapter we explore more options to reduce this cost by adapting the forwarding mechanism of the Interests in the network.

5

Improving Forwarding Techniques

5.1 Introduction

In Chapter 4, we presented the novel protocol MICN that integrates network coding and NDN. It enables NDN to take advantage of network coding and allows fast retrieval of content. MICN enables NDN to overcome the challenges of network coding integration by proposing an Interest naming scheme ensuring retrieval of linearly independent content with each Interest.

Our main objective in MICN is to maximize throughput, *i.e.,* sending content from sources to clients in the least amount of time. As observed for MICN and some other protocols [68] that achieve high throughput with network coding and ICN/NDN, this performance comes at the cost of some flooding of Interests. The flooding results in lingering Interests that remains pending in the network after the clients that sent them have received the corresponding data. These lingering Interests create a notable amount of redundant data flowing in the network (see Section 4.7). This degrades performance in real multi-user, multi-content scenarios. The performance degradation has been observed in the basic operation of sequentially retrieving multiple generations.

Interest cancellation presented in Chapter 4, is the first solution to reduce the impact of the lingering Interests in the network. IC works on the principle of initial flooding and then requires feedback from the clients for the network to identify the lingering Interests and ultimately cancel them. In this chapter, we present alternative approaches to reduce the number of these lingering Interests and consequently the redundant traffic. The alternative approaches are modified Interest forwarding strategies implemented at the clients or the intermediate nodes in the network. These solutions rely on heuristics.

The chapter is organized as follows: we present the effect of lingering In-

terests on the performance in multi-generation content retrieval in Section 5.2. Then several modified Interest forwarding strategies are presented in Section 5.3. A detailed mechanism and forwarding algorithm for each modified forwarding strategy is presented, and the results are compared to the classical MICN (without IC).

5.2 Effects of Lingering Interests in MICN

MICN achieves maximum throughput at the cost of some redundant data traffic that flows in the network. This traffic is a consequence of the Interests that are flooded into the network early in content retrieval. Although MICN achieves the goal of reaching the max throughput, it is still important to consider that this redundant content takes up network resources useful for other content and clients.

The inefficiencies observed are increased download time when another content or multiple successive generations of the same content are requested. The Interests for the next content that arrive later; are processed after the old lingering ones (still in PIT as pending Interests). The nodes process PIT Interests in FIFO order, so older Interests are satisfied before the new ones based on the content available in the CS. To assess the effects of redundant traffic on network performance, we present an example of a network in which clients download multiple successive generations.

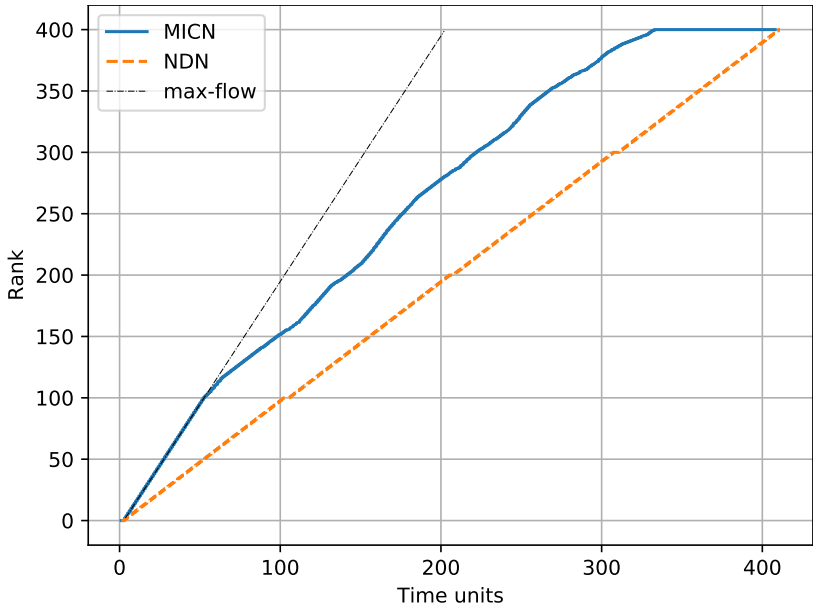


Figure 5.1 – Evolution of the rank of clients[1] in multi-generation content retrieval with NDN and MICN, 4 generations of 100 content segments each w.r.t. simulation time.

Figures 5.1 and 5.2 illustrate the average time required to get back all the content at both the clients and the total data traffic generated in a multi-generation content retrieval on the butterfly topology represented in Figure 4.13.

[1] average of 2 clients in butterfly topology

The simulation parameters and settings are the same as in MICN (Section 4.7.1); however, instead of requesting one generation of the content, the clients request 4 generations of 100 segments successively.

In the butterfly topology (Figure 4.13) the client's max-flow is 2 Data packets per unit time. Figure 5.1 represents the content retrieval time at the client nodes with MICN and NDN. From Figure 5.1, we can observe that only the first generation (rank=100) is retrieved at maximum speed with MICN. During the retrieval of the first generation (time 0 to 50), there is no redundant content. Redundant (non-innovative) coded Data packets start to appear after the complete download of first generation at both clients (after time 50), as shown in Figure 5.2. This happens because coded data for the first generation continues to exchange between some nodes due to lingering unsatisfied Interests. These Data packets take up the network resources causing the delay in download for the next requested generations.

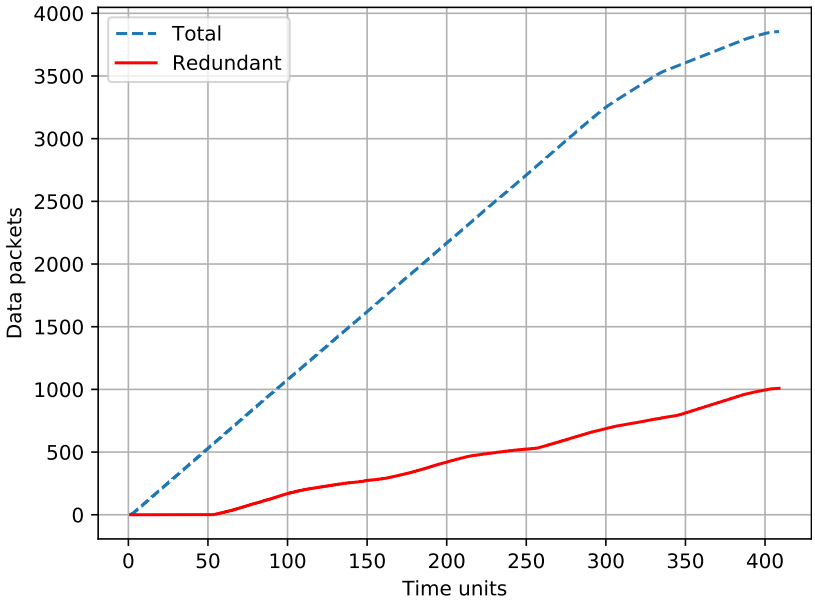


Figure 5.2 – Count of total and redundant (non-innovative) coded Data packets exchanged in the entire network in multi-generation content retrieval with MICN, 4 generations of 100 content segments each, w.r.t. simulation time.

One Interest might sometimes bring multiple innovative coded data from different sources for maximum throughput, requiring first a forwarding on several faces. The issue then is that the clients' Interest generation rate proves ultimately too high, resulting in excess, lingering Interests as in our scenario. The related general issue (appearing in the butterfly topology) is that some nodes' max-flow is different from others. The content retrieval rate of some nodes is slow, and the additional data traffic due to the lingering Interests, although not relevant for the actual client, will still be innovative for these nodes with a slower download rate. This additional data might be useful if extra clients were present.

Nevertheless, adjusting Interest generation and modifying the Interest forwarding in the network are possible solutions to reduce the extra traffic. We proposed an Interest cancellation technique in Section 4.5.2 to limit the data

traffic generated by the lingering Interests. Interest cancellation requires a feedback mechanism and cancels the Interests in the network for content that the client no longer requires.

In the following section, we propose additional methods to reduce the lingering Interests that can be implemented at the nodes to limit the number of lingering Interests with MICN. These methods provide a decentralized solution without the need for a feedback mechanism.

5.3 Modified Forwarding Techniques

MICN, as described in Chapter 4 uses a multicast forwarding strategy where the nodes forward Interests to all the available faces. Using this forwarding strategy helps the Interests to reach all the sources by taking advantage of multiple paths in the network. With network coding, the same Interest can result in different coded contents when different copies of that Interest reach different caches/sources.

MILIC Interests ensure that linearly independent content is received with each distinct Interest. Additionally, there is a high probability that two MILIC Interests with the same index bring linearly independent content. For example, in the butterfly topology (Figure 4.13) the client U_1 and U_2 send each Interest I_i with $i = 1, \ldots, n$ on their two available faces. A linearly independent content is received for each Interest I_i from both faces. It was observed that most of the content was retrieved at the clients from Interests for initial indices (1 to $\approx n/2$). This is the direct consequence of Property 4.3 of MILIC subsets, *i.e.,* two linearly independent contents for the same index yield two contents for different indices after Gaussian elimination. The router of the client node received 2 linearly independent contents for one index, as each index was forwarded to different faces. The Interests for the following indices that the clients send with each incoming content brought content, but it eventually becomes redundant for the client nodes.

With MILIC with Interest cancellation, the Interest generation and forwarding strategy are kept the same as in plain MICN, and some Interests are canceled based on the content received at the clients.

A natural approach to reducing this traffic would be to change the forwarding strategy and use a forwarding strategy that only forwards Interests to one path. Replacing the multicast strategy can help reduce redundant traffic by sending fewer Interests. Nevertheless, it no longer allows to take advantage of the multi-source and multi-destination, and consequently the fast retrieval of content can no longer be achieved. Indeed a multicast strategy is required to achieve max flow with network coding.

In the following sections, we present some time-varying forwarding methods that change dynamically over time. The main idea is to let the node use the multicast strategy at the beginning in order to take advantage of the multiple available paths. Then based on the rate of content retrieval and the state of caches, the forwarding strategy can be changed (sending on one face or not forwarding the Interests at all). The decision of changing forwarding strategy over time is taken considering the state of the nodes and clients.

Forwarding Technique	Implemented at	Network Parameter
Threshold	intermediate nodes	queue lengths
Short Queue	intermediate nodes	queue lengths
Dynamic pipeline	clients	pipeline size
Replication factor	intermediate nodes	no. of faces
Expected rank	intermediate nodes	no. of forwarded Interests+ rank of the cache

Table 5.1 – List of different Forwarding Techniques

We propose different methods considering different parameters. We explore the design space for such methods and design mechanisms for the implementation of these algorithms. We also present insights into the choice of these methods. Simulations are performed, and an in-depth analysis of the behavior of nodes with each method is presented. The modified techniques are then tested on the butterfly topology (See 4.13), which is clearly an unrealistic network however this simple network helps illustrating and understanding of the improved forwarding approaches. To verify the results with the modified strategies they are additionally tested on a complex more realistic PlanetLab Topology (See 4.20)

We study in detail the information available at the nodes and clients to help the nodes take a different forwarding decision. Testing these approaches helped us study the network behavior and the effects of implementing varying forwarding techniques on the content delivery time and the cumulative data traffic inside the network with MICN.

Table 5.1 presents a list of the proposed methods and the decision parameters. The following sections present detail of the proposed methods and the modified forwarding algorithms implemented at the nodes. The proposed forwarding techniques are tested on the butterfly topology and the PlanetLab topology. The simulation parameters are the same as in MICN (Section 4.7.1), where clients request one generation of content. We compare the results with normal MICN (without IC).

5.3.1 Threshold-Based Forwarding

To avoid lingering Interests in the network, the number of Interests forwarded in the network needs to be limited. An approach for this would be to limit the number of pending Interests forwarded in the network. We propose to do so by putting a limit on the number of Interests a node forwards.

The PIT stores the information of pending Interests queue lengths, *i.e.,* the number of Interests a node has forwarded on its faces. With the basic MICN forwarding, the nodes replicate incoming Interests to all available outgoing faces without a limitation. We propose a *threshold-based forwarding* (TBF) technique that limits the queue length on the faces of a node. This limit is added by defining a threshold θ for the queue length at the node. This technique is easy to implement since each node has the information of the pending queues on its faces readily available.

With the threshold-based forwarding, the nodes continue with the multicast

strategy as MICN but stop forwarding more Interests on a face as soon as the queue length of a face reaches the threshold θ. However, the Interest generation at clients remains the same as MICN, *i.e.*, after the initial pipeline burst, an Interest for a new index is sent as soon as linearly independent content is received.

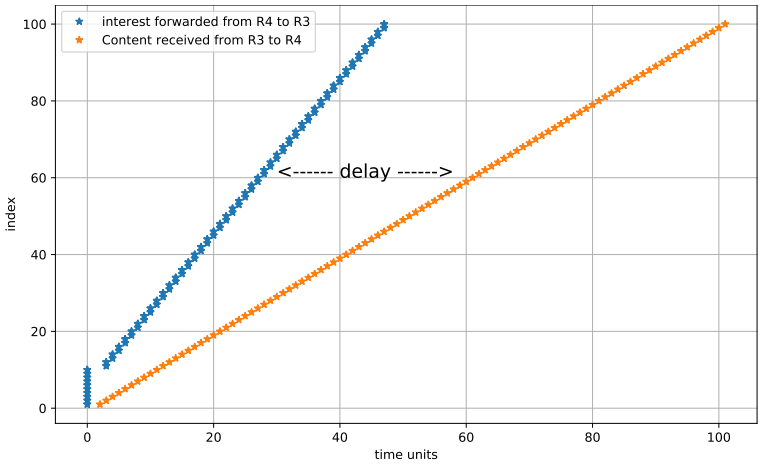


Figure 5.3 – Forwarding time of Interest and arrival time of content for node R4 in butterfly topology with MICN

The main idea is that forwarding more Interests on faces with long queues results in increased queue lengths but does not affect the rate at which content flows back. Figure 5.3 presents the forwarding time of Interests and the arrival time of content at the node R_4 of the butterfly topology with MICN. Figure 5.3 presents the cumulated Interests from both the clients U_1 and U_2 at the node R_4. Node R_4 forwards the Interests from the clients for the generation of size 100 around time unit 50. The rate at which the content is received by the node R_4 is much slower because its max flow is 1 compared to 2 for the client nodes. The content continues to flow until time 100, while the content at the clients is received by time 50. The same happens on other nodes of the network with lower max flow. The distance between the Interest forwarding time and content

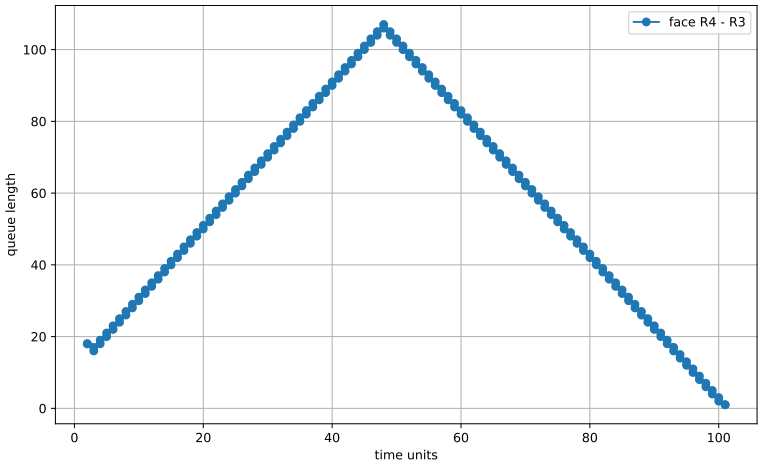
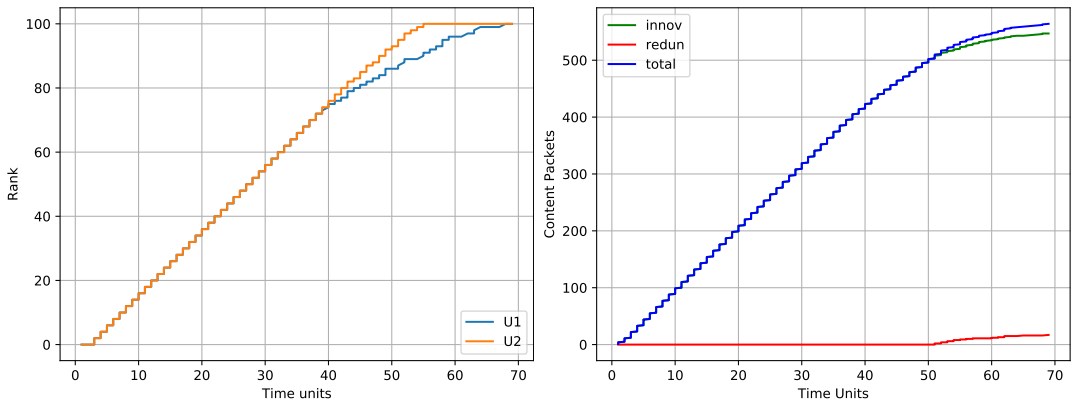


Figure 5.4 – Pending Interest queue length at R_4 towards R_3 in butterfly topology with MICN

arrival time in Figure 5.3 represents the delay in content arrival. The delay is increasing with time.

Figure 5.4 shows the queue of pending Interests at R_4 towards the node R_3. The queue length increases until all the Interests from the clients have been forwarded. The Interest propagation starts at time zero by the clients. The pending queue length at node R_4 becomes 20 immediately (with some propagation delay) due to the initial burst of Interest in MICN from both clients U_1 and U_2. As observed in the butterfly topology results (see Section 4.7.2, Figure 4.14b) the entire generation is received at the client at a time less than 50 time units. The clients do not send Interests once the complete generation is downloaded. Nevertheless, the nodes are continuing to forward the Interests because they are unaware that the client has received the required content. The queues then continue to decrease, indicating the content is still flowing in response to these Interests.

The threshold limits queue length on each face of the node to limit the forwarding. A node forwards an Interest only if the queue lengths are shorter than the threshold. This strategy sends on all available faces until the set threshold is reached. Later it chooses to send on one or more faces based on their queue lengths. If only one face has a queue less than the threshold, the node forwards Interest on that one face. Otherwise, if multiple faces have queues that are not full the node forwards on all these faces. The nodes in the network discard the Interests if none of their faces have a queue size less than the threshold. We tested this technique with a fixed threshold initially equal to the pipeline size (10). The initial pipeline of Interests is multicast in the network, and afterward, the forwarding depends on queue lengths compared to θ. The clients only resend Interests for missing indexes after the initial Interests time-out.

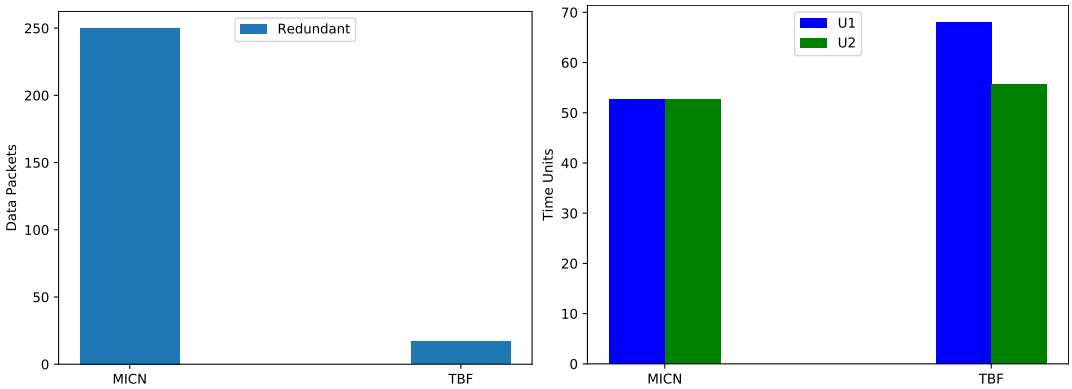


(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.5 – Results of the threshold-based forwarding strategy in butterfly topology ($\theta = 10$)

The modified strategy was tested on the butterfly topology in Figure 4.13 with the same simulation parameters of MICN (see Section 4.7.1). Figure 5.5 presents the results of the threshold-based forwarding technique in terms of download time and the total content traffic in the network. Figure 5.6 com-



(a) Redundant data traffic (b) Content retrieval time

Figure 5.6 – Results of the threshold-based forwarding (TBF) strategy in butterfly topology ($\theta = 10$)

pares the results with classical MICN in terms of redundant traffic and content retrieval time. It can be observed that limiting queue lengths presented promising results in terms of the redundant content, and consequently, total content traffic in the network was reduced considerably. Figure 5.6a indicates the reduction of redundant data traffic, however, max throughput is no longer achieved at the *clients* (see Figure 5.6b).

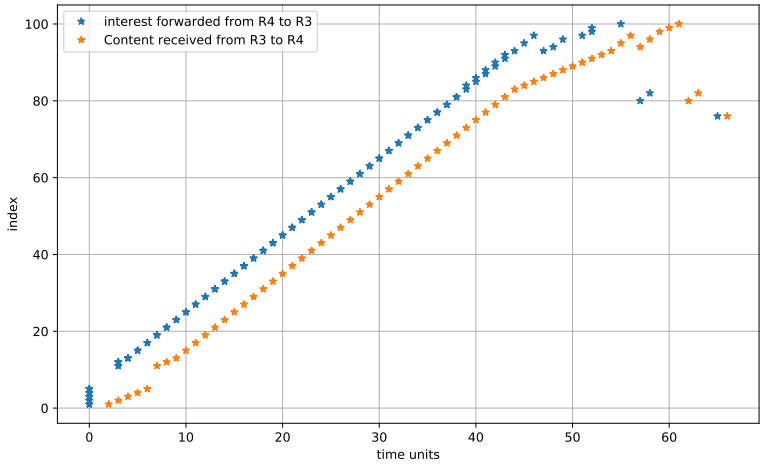


Figure 5.7 – Forwarding time of Interest and arrival time of content for node R4 with threshold-based forwarding in butterfly topology ($\theta = 10$)

Max flow was not achieved using threshold forwarding. Nevertheless, the results present valuable insights on the Interest forwarding in the network. Multiface forwarding is beneficial for fast content retrieval; however, the delay between the Interests forwarded and content received should be considered when forwarding Interests. The rate of Interest forwarding does not reduce the time of arrival between two content packets.

Figure 5.7 presents the Interest forwarding and content arrival time with the threshold-based forwarding at the node R_4 (butterfly topology). We observe that the delay between the Interest forwarding time and content arrival time becomes smaller. The nodes discard Interests received after the queue threshold.

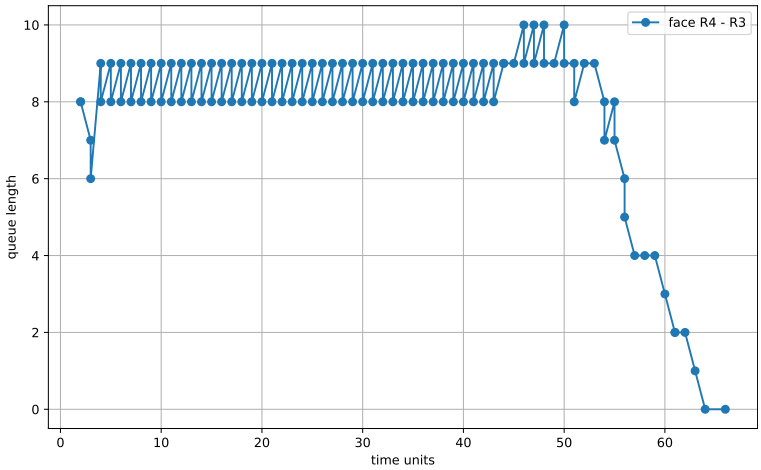
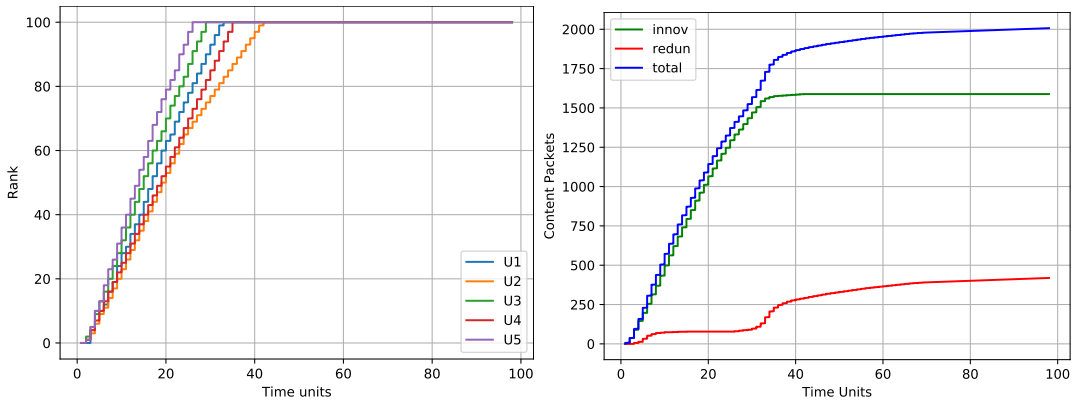


Figure 5.8 – Pending Interest queue length at R_4 towards R_3 when threshold-based forwarding strategy is used in MICN in butterfly topology ($\theta = 10$)



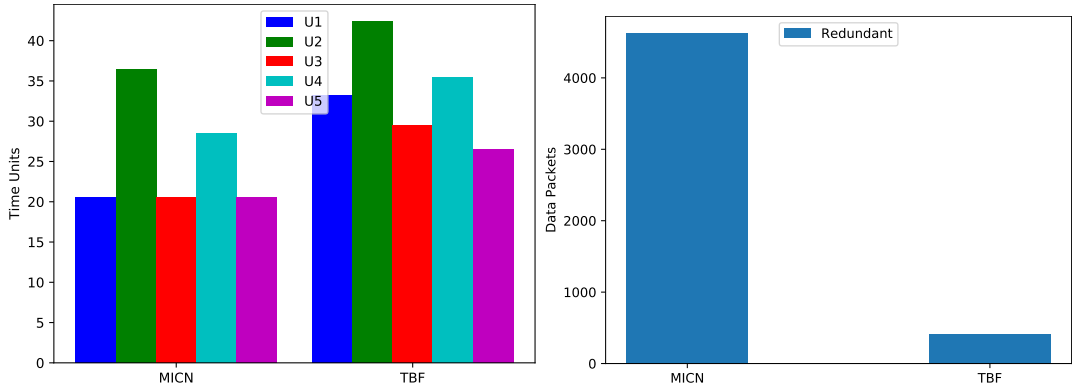
(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.9 – Results of the threshold-based forwarding strategy in PlanetLab topology ($\theta = 12$)

The clients do not receive content for some of the discarded Interests, hence the delay in content retrieval time (Figure 5.5a). The clients then have to re-request missing indices after the Interest for these indices time-out. In Figure 5.7 the re-requested indices appear after time 60. We observed that by reducing the rate of Interest forwarding, we prevent long queues from developing on outgoing faces Figure 5.8. There are fewer lingering Interests and consequently less redundant data traffic.

Figures 5.9a and 5.10b present the results with the threshold-based forwarding on the PlanetLab topology with $\theta = 12$. Figures 5.10a and 5.9b present a comparison in the content retrieval time and the redundant traffic in the network with classical MICN. The redundant data traffic is reduced to 9.07% of the redundant traffic generated with MICN.



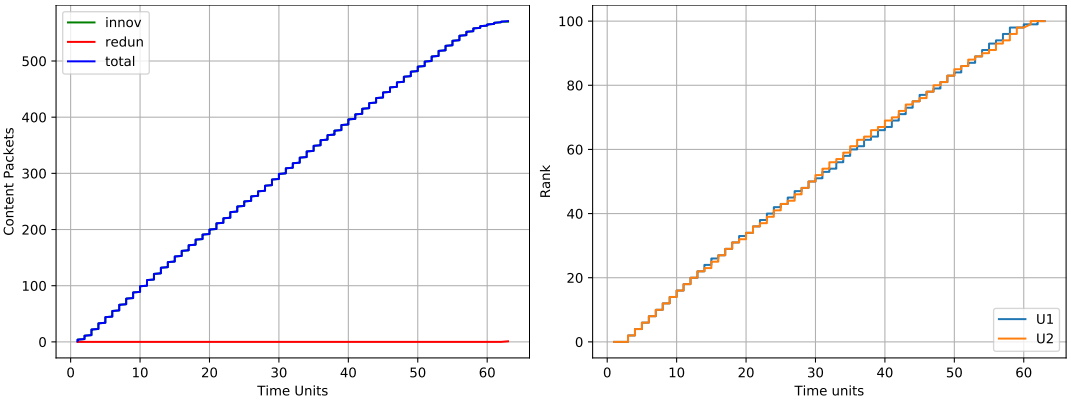
(a) Content retrieval time (b) Redundant data traffic

Figure 5.10 – Results of the threshold-based forwarding (TBF) strategy in PlanetLab topology ($\theta = 12$)

5.3.2 Forwarding on Shorter Queue

The results with the threshold-based forwarding are promising in solving the redundant content problem in the network. These results lead us to explore other methods that reduce the number of Interests forwarded in the network. One approach is to choose faces to forward an Interest based on their pending queues lengths.

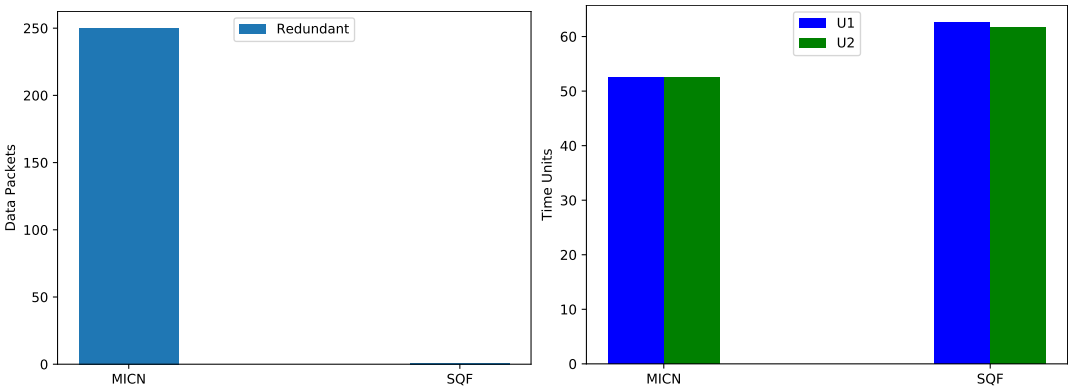
We propose a *shorter queue-based forwarding* (SQF) that choses the face with the shorter queue after the first burst of Interests in MICN. Sending Interests only on the face with a shorter queue length helps avoid sending Interests on slower faces or with already long queues of Interests. Forwarding on less busy faces helps to avoid over-occupying links with Interests that will either expire or bring back content later than required. It also mimics the same effect as threshold-based forwarding but allows the nodes to send at least one Interest each time an Interest is received.



(a) Evolution of cumulative data traffic as a function of time (b) Rank evolution as a function of time

Figure 5.11 – Results of the shorter queue-based forwarding approach

The Interest generation strategy is the same as MICN. The strategy excludes the first burst of Interests in MICN. A multicast strategy for the initial burst of



(a) Redundant data traffic (b) Content retrieval time

Figure 5.12 – Results of the shorter queue-based forwarding (SQF) approach

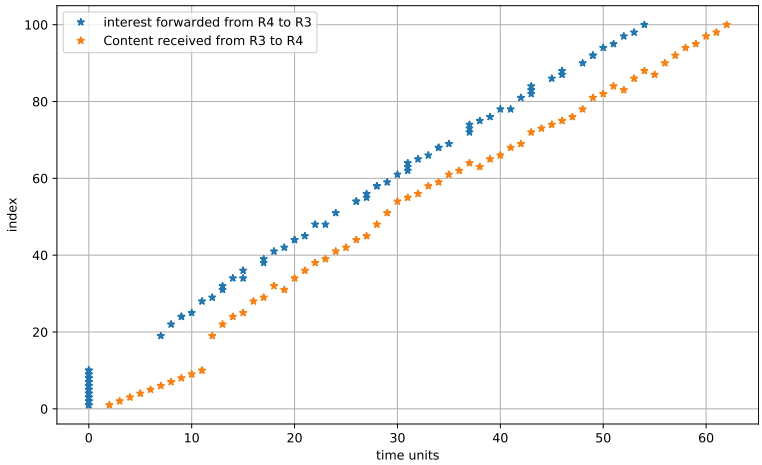


Figure 5.13 – Forwarding time of Interest and arrival time of content for node R4 with shorter queue-based forwarding

Interests helps to get a continuous flow of Data packets (similar to the threshold-based forwarding in Section 5.3.1).

The shorter queue-based strategy is tested on the butterfly topology (see Figure 4.13). The results show that multicasting the first burst of Interests allows the Interest to reach the sources faster, and hence the rate of content retrieval is fast. Later the Interests start to flow slowly; hence the content flow slows down as well. The total redundant content with this strategy is negligible (see Figures 5.11a and 5.12a). The content retrieval speed does not reach the maximum value (see Figures 5.11b and 5.12b). However, it is faster than the threshold method since Interests are not discarded but only limited in forwarding in the network by forwarding once (on one face with shorter pending Interest queue) only.

Figure 5.13 indicates that the delay between Interests and content is reduced compared to normal MICN forwarding in Figure 5.3. The queue length represents the fact that the initial burst of Interests is allowed to forward without limit on flooding, and node R_4 in butterfly topology forwards the Interests of both the clients U_1 and U_2 see Figure 5.14.

Figures 5.15 and 5.16 presents the result with the shorter queue forwarding

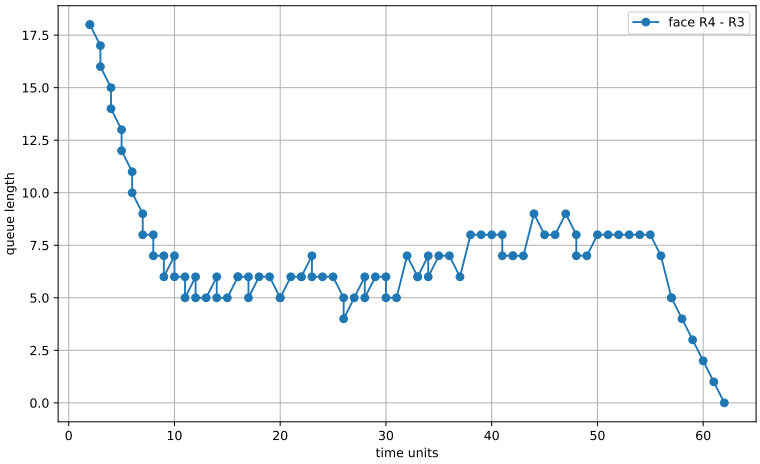
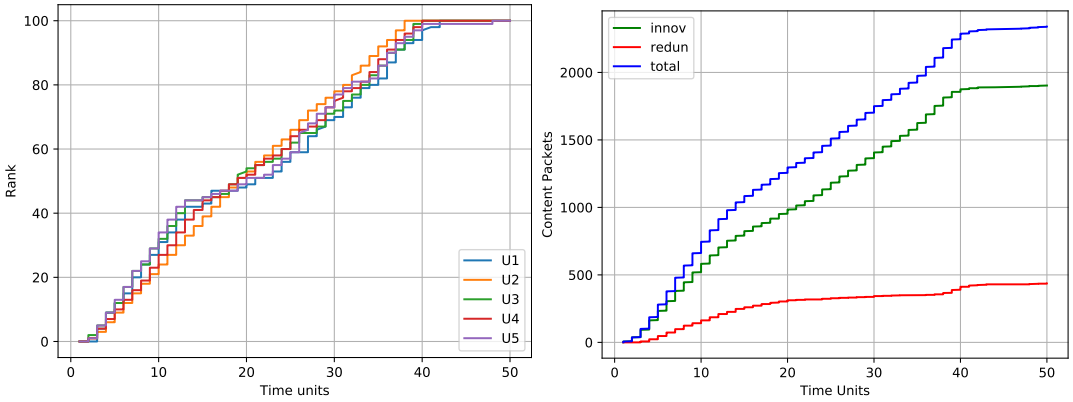


Figure 5.14 – Pending Interest queue length at R_4 towards R_3 in shorter queue-based forwarding approach

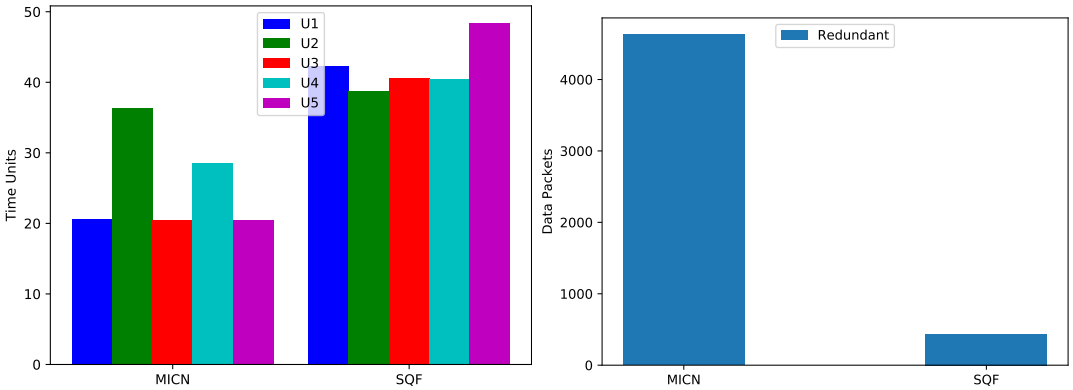


(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.15 – Results of the shorter queue-based forwarding approach

strategy on the PlanetLab topology. The content retrieval time at the clients is slower than the with MICN; however, the redundant traffic is a mere 9.4% of the redundant traffic with MICN (see Figure 5.15).



(a) Content retrieval time

(b) Redundant data traffic

Figure 5.16 – Results of the shorter queue-based forwarding (SQF) approach

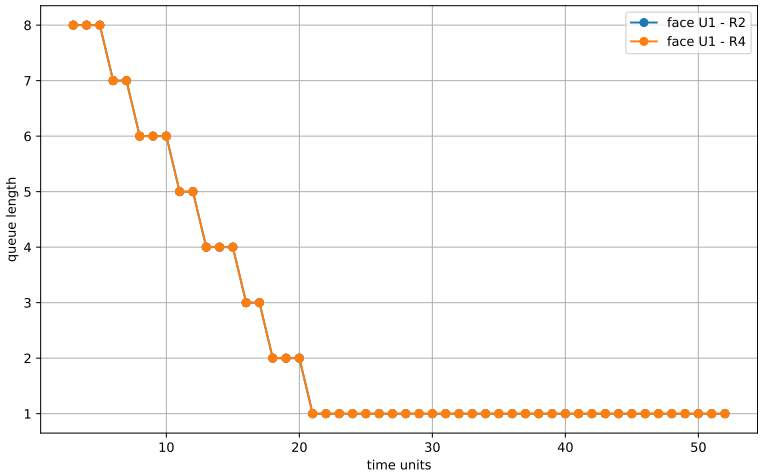


Figure 5.17 – Evolution of pending Interest queue length at client U_1 in the gradually decreasing pipeline approach with $\alpha = 2$

5.3.3 Gradual Decrease of the Pipeline Size

MICN allows the client nodes to send multiple Interests at the beginning of content retrieval. This initial burst enables the clients to send enough Interest in the network to have a continuous flow of content when the content starts flowing back. The clients in MICN then maintain a constant pipeline size during the content retrieval. That means that as long as the entire generation is not received, the the clients have a number of pending Interests equal to the pipeline size. They ensure that the flow of content is continuous, with clients having pending Interests at all times. Given a sufficient pipeline size, MICN can reach the maximum capacity as detailed in Section 4.7 (see Figures 4.18 and 4.25).

Nevertheless, we observed that clients keeping a constant pipeline size throughout the content retrieval and multicast forwarding strategy at nodes results in Interest generation to be ultimately too high. For example, in butterfly topology, the Interests 0 to 50 are responsible for most clients' content as they send Interests on two faces and receive two content packets for each. This is a direct consequence of the MILIC properties 4.2 and 4.3.

This observation leads us to explore the possibility of controlling the rate at which the clients generate Interests. With the threshold-based and shorter queue-based forwarding, we controlled forwarding strategy at the nodes; we now propose controlling Interest generation at the client. The main idea is to have the same pipeline size at the beginning of content retrieval to send enough Interest required for the continuous flow of content. Then gradually reduce the size of the pipeline maintained by the clients. Clients dynamically reduce pipeline size based on the amount of content they have received. We denote this *gradually decreasing pipeline* (GDP) strategy.

The clients now maintain a gradually decreasing size of the pipeline p_t at any time t (per generation). We propose a linear decrease based on the current pipeline size p_t, the rank received at a client's cache R_t, the size m of the generation g of the content being requested and a scaling factor $\alpha > 0$. The pipeline size is then computed as follows

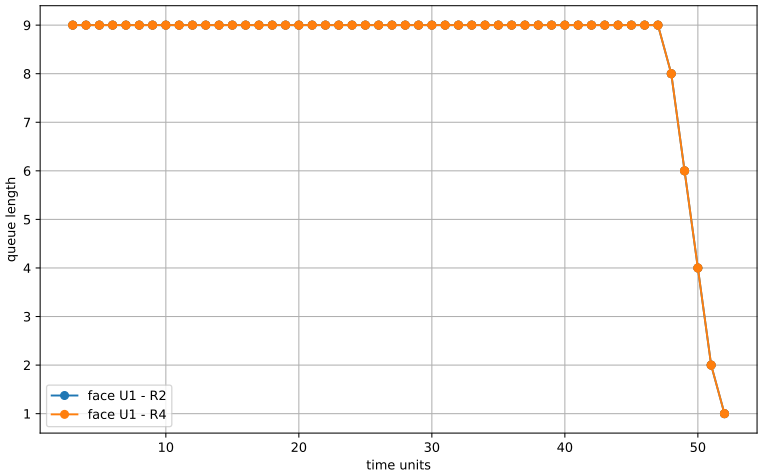


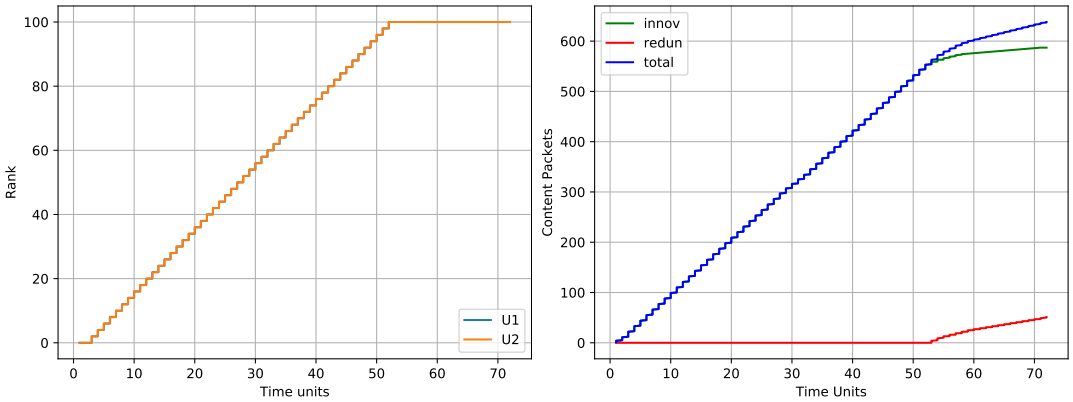
Figure 5.18 – Evolution of pending Interest queue length at client U_1 in MICN with constant pipeline

$$p_{t+1} = \max\left(1, p_t(1 - \alpha\frac{R_t}{m})\right). \tag{5.1}$$

When $\alpha = 0$, the pipeline size does not change, which is equivalent to plain MICN. The rate at which the size reduces depends on α. The larger the value of α, the faster the size reduces with each data received at the client. Equation 5.1 ensures that $p_t > 1$, *i.e.,* there are always outstanding Interests until the entire generation is received at the client. The experiment done to test the reducing pipeline window over the butterfly topology, has a variable value of scaling factor α.

Figure 5.17 presents the gradually decreasing queue lengths at the client U_1 with $\alpha = 2$. The queue length of the client U_1 in MICN (Figure 5.18) stays constant until all the indexes in a generation are requested. After all the indexes are requested, the queue lengths start to decrease.

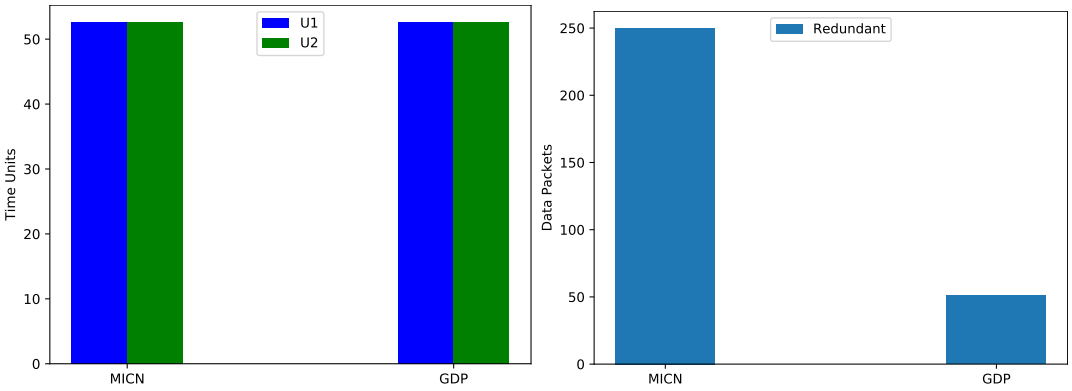
Figure 5.19a and 5.19b presents the results obtained from the experiment regarding the time of rank retrieval and the total data in butterfly topology. Figure 5.20a show the content retrieval with this strategy is near max flow.



(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.19 – Results of gradually decreasing pipeline size with $\alpha = 2$



(a) Content retrieval time (b) Redundant data traffic

Figure 5.20 – Results of gradually decreasing pipeline size according to 5.1 (GDP) with $\alpha = 2$

Figure 5.21 presents the redundant traffic with the varying value of α. Higher values of α mean fast reduction of the pipeline size, reducing the number of Interests sent in the network, consequently reducing the redundant traffic. The experiments show that if $0 \leq \alpha \leq 2$, the content retrieval time is close to the plain MICN (see Figure 5.22). However, the traffic reduces considerably with increasing value of α (Figure 5.21).

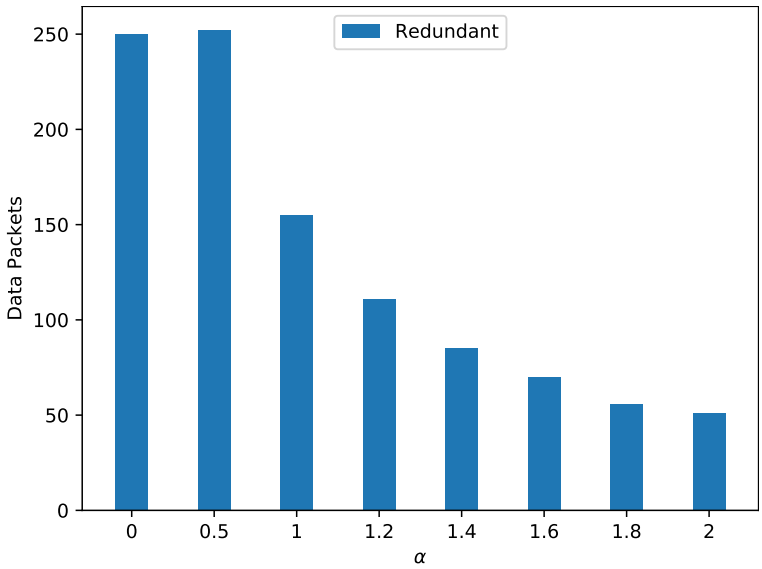


Figure 5.21 – Redundant data traffic in butterfly topology with the gradual decrease of pipeline with different values of α

This strategy indicates that indeed the Interest generation rate in combination with the multicast forwarding is high in MICN. The clients can thus reduce their pipeline size and still get back the requested data fast enough while avoiding redundant data.

The gradual reduction of the pipeline is tested over the PlanetLab topology. The results in terms of time and data are promising presented in Figures 5.23 and 5.24. The α parameter is tuned differently for PlanetLab topology with 5 clients with different throughput, min-cut, and distance to the source.

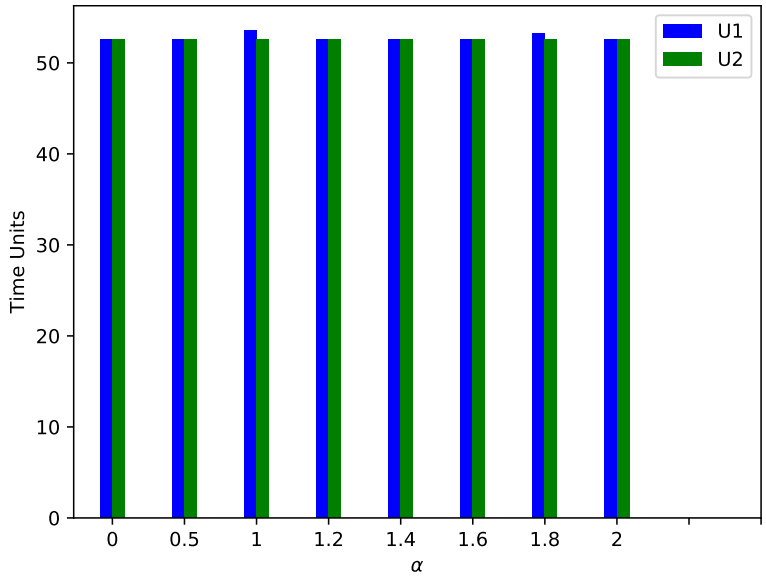


Figure 5.22 – Content retrieval time in butterfly topology with the gradual decrease of the pipeline size with different values of α

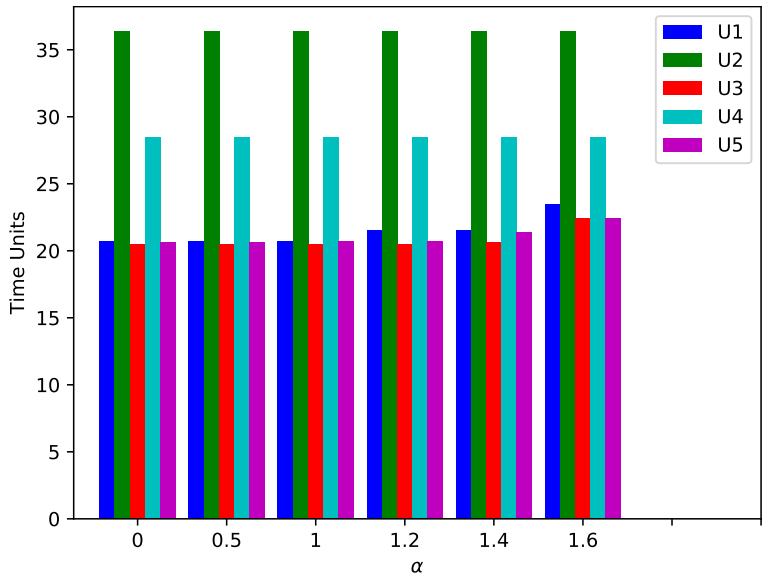


Figure 5.23 – Content retrieval time in PlanetLab topology with the gradual decrease of the pipeline size with different values of α

In conclusion, reducing pipeline size at the client is efficient in terms of data retrieval time and total data traffic. However, it is essential to note here that it is possible to correct the factor α at the clients. The client application that generates Interests should tune α. However, the application receives one content for each Interest packet it generates since the replication of Interests happens at the nodes in the network. The application does not have the information if multiple packets are received for one Interest. Some feedback mechanism from the node is required for the application to tune the factor α. If a node has an estimate of how fast content comes in, it can slow down the replication of Interests. This is similar to reducing the pipeline size locally, instead of at the clients.

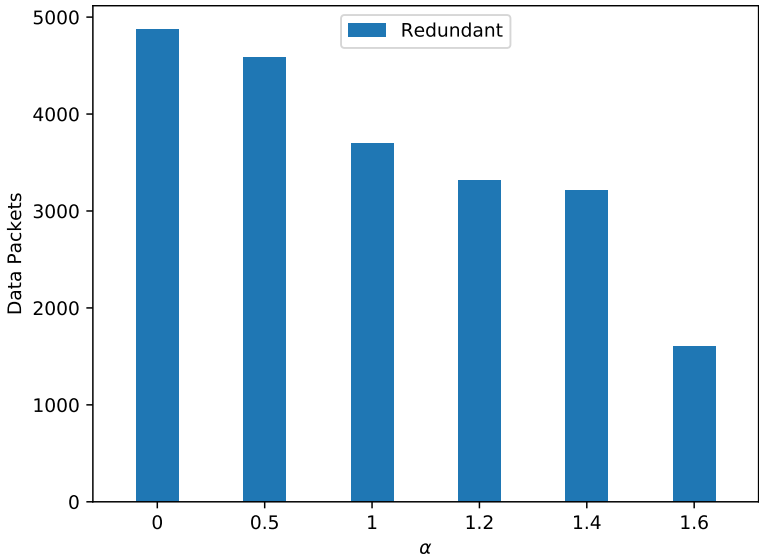
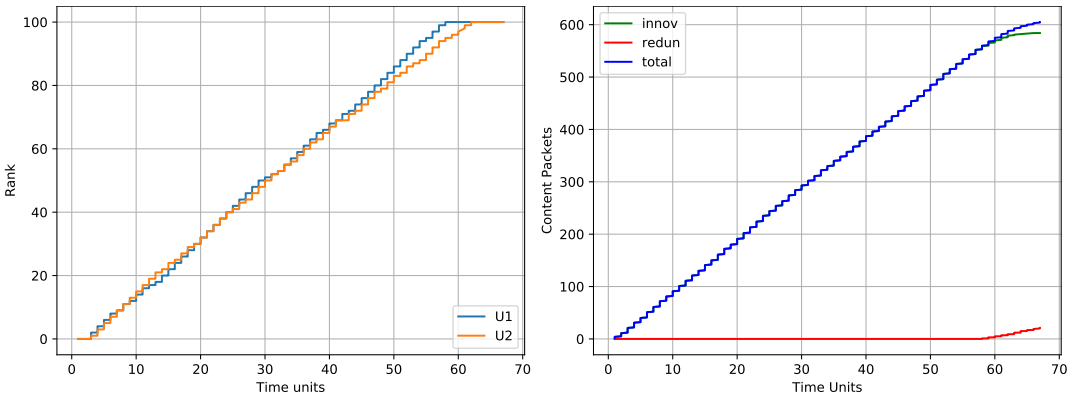


Figure 5.24 – Redundant data traffic in PlanetLab topology with the gradual decrease of the pipeline size with different values of α

5.3.4 Replication Factor-Based Forwarding

The multicast forwarding strategy, replicates every Interest on all the available faces of a node. This replication is one reason for the excessive number of Interests circulating in the network. One of the methods to tackle the excessive number of Interests is to control the replication of Interest at the nodes. We propose control this replication by implementing a replication probability at the nodes. The main idea is to let the nodes choose between sending Interests on all outgoing faces and sending on some faces randomly with a certain probability. Unlike the techniques in Section 5.3.1 and Section 5.3.2, there is no threshold or limit for the node to reach before they start modifying their forwarding technique.

With this method, instead of forwarding to all faces (multicast strategy), the

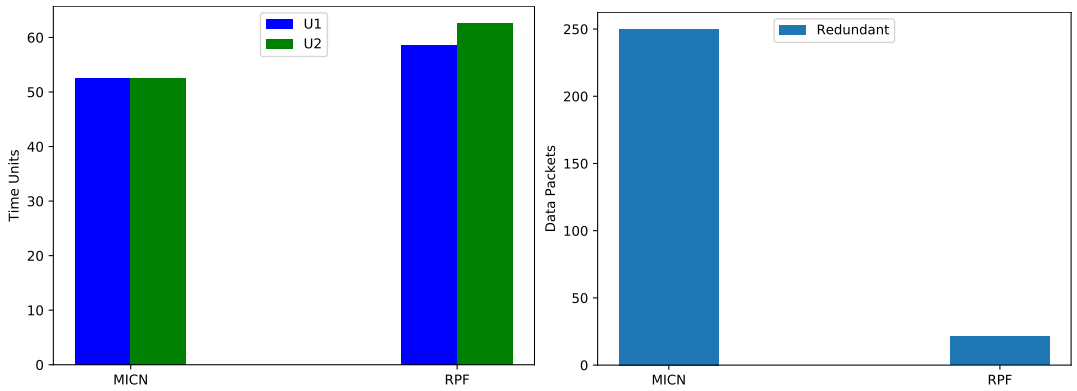


(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.25 – Results of Replication Factor-based approach on butterfly topology with $p_r = 30\%$

nodes have a varying forwarding strategy. For example, a node with two outgoing faces can choose to send on two faces or one face with a certain probability. The decision to replicate the Interests is based on a replication probability p_r and hence called *replication factor-based forwarding* (RPF). To test this approach we implemented a constant replication probability p_r on all the nodes of a network. Each node in the network replicated the Interests received with $p_r = 30\%$, *i.e.,* each node replicated 30% of the Interests it received to all its faces. The rest of the Interests are forwarded only to one face chosen randomly.



(a) Content retrieval time (b) Redundant data traffic

Figure 5.26 – Results of Replication Factor-based forwarding (RPF) on butterfly topology with $p_r = 30\%$

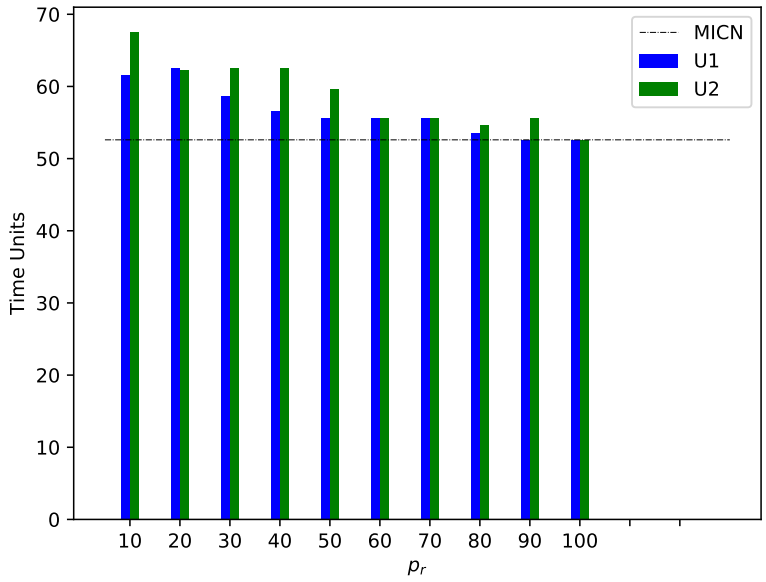


Figure 5.27 – Content retrieval time in butterfly topology with with different replication probabilities p_r

Figures 5.25a and 5.25b presents the results obtained from the experiment in terms of the time of rank retrieval and the total data in butterfly topology. Figure 5.26b shows that reducing the number of times a node replicates Interests reduces the total data traffic in the network. The download time increases

compared to the multicast strategy of MICN (see Figure 5.27) since the forwarding choice is random in terms of the number of faces and the faces chosen for forwarding. However, the overall redundant data can be reduced to less than 20 packets compared to 250 with MICN.

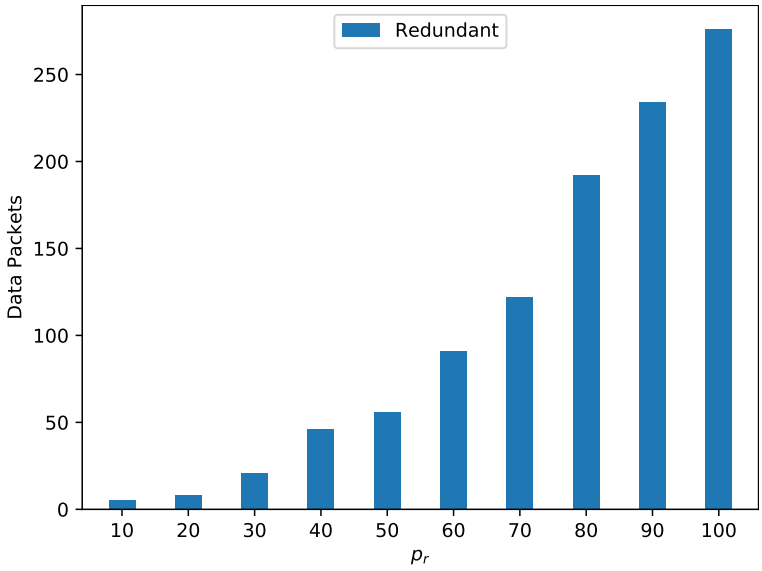


Figure 5.28 – Redundant data traffic in butterfly topology with different replication probabilities p_r

Figures 5.27 and 5.28 presents the effects of changing the replication probability at the nodes. If $p_r = 100$, *i.e.,* all nodes replicate Interests on all available faces 100% of the time, and it gives the same results as MICN. Also, $p_r = 100$ generates the most traffic. Decreasing the value of p_r decreases the redundant traffic but increases the content retrieval time.

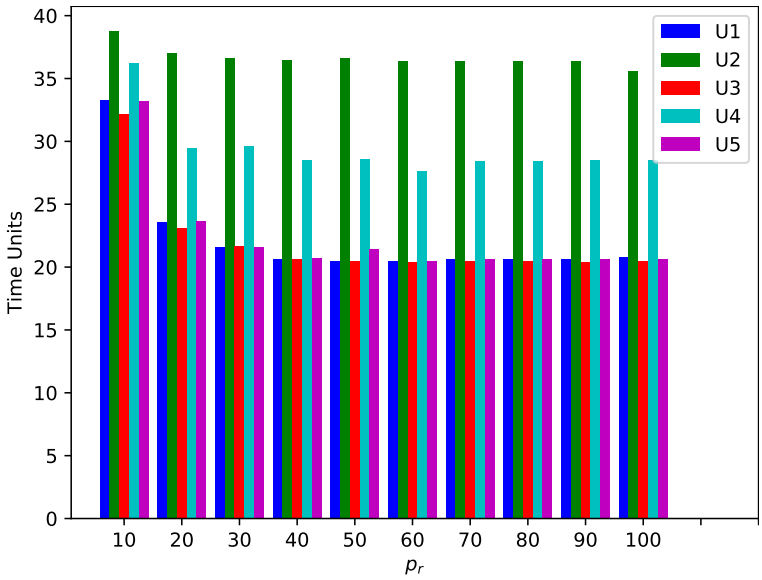


Figure 5.29 – Content retrieval time in PlanetLab topology with with different replication probability p_r

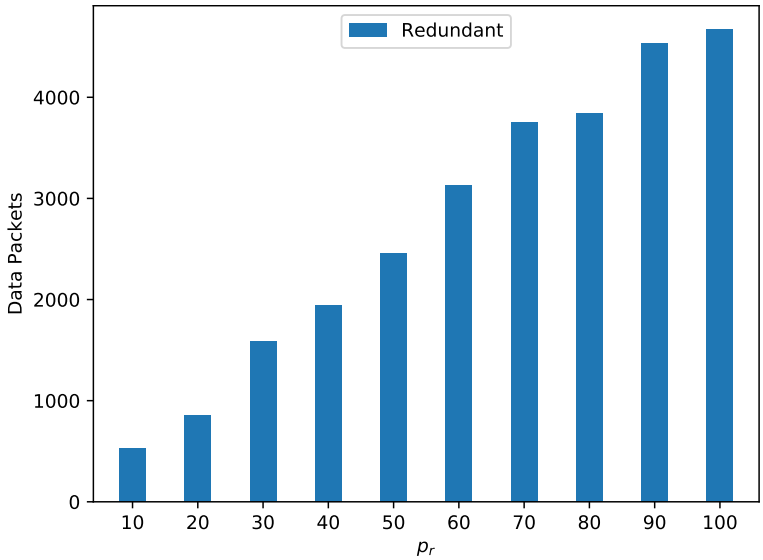


Figure 5.30 – Redundant data traffic in PlanetLab topology with different replication probabilities p_r

Figures 5.30 and 5.29 presents the result with replication factor-based strategy on the PlanetLab topology. The results with the PlanetLab topology show similar trends as the butterfly topology in redundant traffic. The content retrieval time, however, becomes close to MICN with a replication factor of 60%. This indicates that different networks with different numbers of clients and different network parameters such as connectivity, throughput, *etc.*, require the replication factor to be tuned differently. It is important to note here that the replication factor implemented in the example scenarios stays constant throughout the content retrieval. However, a dynamic modification of this replication during content retrieval could improve both content retrieval time and data traffic.

In conclusion, the results with the replication factor strategy are quite promising. Nevertheless, the difficulty is for nodes to decide when and how much they need to adjust their replication factor.

5.3.5 Expected Rank-Based Forwarding

By studying the methods presented in Sections 5.3.1-5.3.4 we can identify the actions taken at the nodes to solve the redundant data problem. We observed that all the presented methods work efficiently in reducing the redundant data in the network. The maximum throughput, however, is not achieved with all of them. Another critical point that we observed is that dynamic forwarding is an effective tool to reduce redundant traffic; choosing the tuning parameters at each node is not straightforward. All the methods above are tried and tested on butterfly topology with fixed parameters.

Using all the results from the above techniques, we present a technique that uses available information at a node to decide to change the forwarding strategy on the fly. We use this information to predict the effective (data) replication a node has achieved at any time k and the expected rank of a node at time k given the Interests forwarded by the node. We denote this forwarding technique

expected rank-based forwarding (ERF).

The replication achieved r_{ach} at a node is computed by

$$r_{ach} = R_k / i_{max,k} \tag{5.2}$$

where R_k is the rank of the cache at any time k and $i_{max,k}$ is the maximum MILIC index in the cache prior to Gaussian elimination.

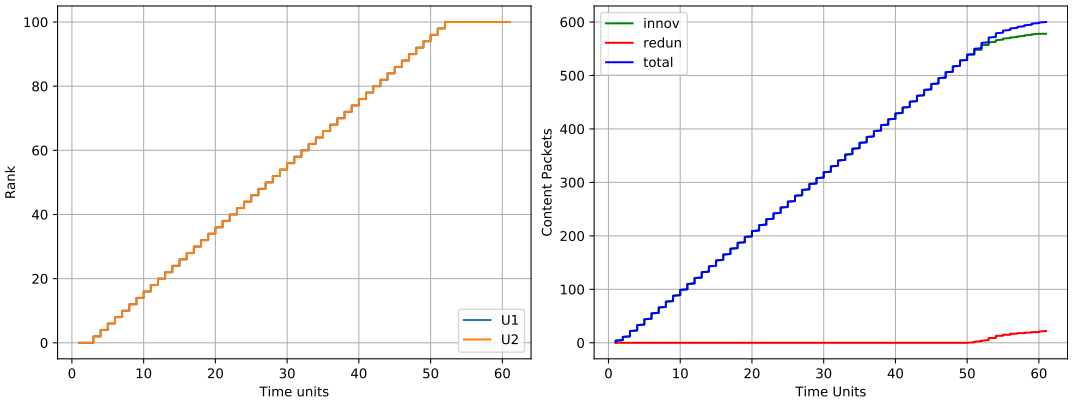
As we know that with MILIC properties 4.2 and 4.3 a node can have rank higher than the max index received. For example, a node with two outgoing faces can have a rank of 4, while the maximum index in the cache is 2. It is possible due to the MILIC properties 4.3 and 4.2, *i.e.*, independent content received in response to Interests for index 1 and 2 can help the node have a rank 4. Here the r_{ach} of this node is 2.

Another information available at the node is the Interests that a node has forwarded. At any time, the achieved replication factor and the forwarded Interests can give an estimate of the expected rank this node (can) achieve after the data for the forwarded Interests is received. An estimate of the expected rank of a node is computed by

$$R_{e,k} = r_{ach} \times i_{f,k} \tag{5.3}$$

where $i_{f,k}$ is the maximum index of the forwarded MILIC Interest in the PIT. We use the maximum index to compute the estimate, considering that MILIC Interests are received in order. (5.3) gives a ratio between the number of Interests forwarded and number of content packets received by the node, this effectively translates into the number of useful interests forwarded or the replication achieved by the node.

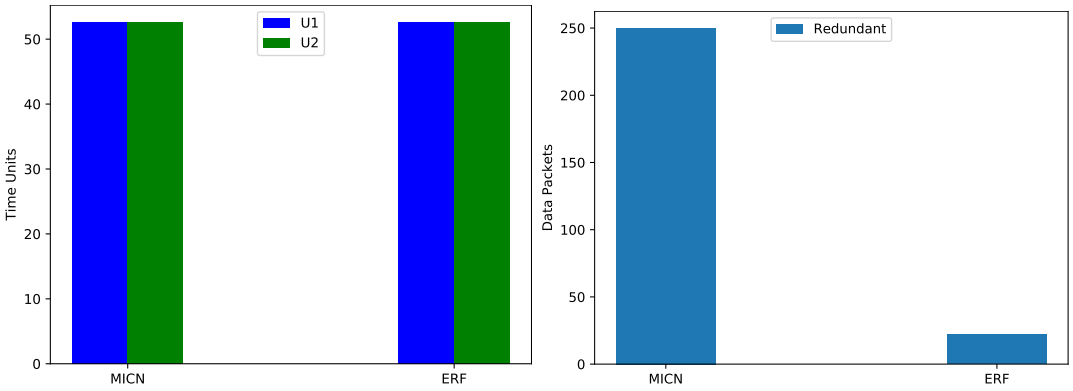
Note that the generation size m is the maximum value a node can expect to receive. We propose that the nodes use this estimate to tune their forwarding strategy. A node should decide to continue to forward the Interests as long as $R_{e,k} \leq m$. At any time this value becomes strictly larger than the generation size m the node should stop forwarding Interests. It is important to note that this value can drop back to less than m if not enough content is received in



(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.31 – Results of expected rank-based approach in butterfly topology



(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.32 – Results of expected rank-based Forwarding (ERF) in butterfly topology

response to the forwarded Interests (see Figure 5.35). As it affects the value of r_{ach} and as soon as this happens, the node can resume forwarding the Interests.

Figures 5.31a and 5.31b present the results obtained from the experiment in terms of the time of rank retrieval and the total data in butterfly topology. This method performs as plain MICN in terms of data download time. The number of redundant Data packets is reduced to less than 25 packets (see Figures 5.32b and 5.32b). This approach is different from the strategies based on the threshold, pipeline, or replication factor. The nodes decide to change their forwarding strategy during the content retrieval with no external parameter (*e.g.*, θ, α or p_r). Here we assume that the generation size m is known at all the nodes in the network[2].

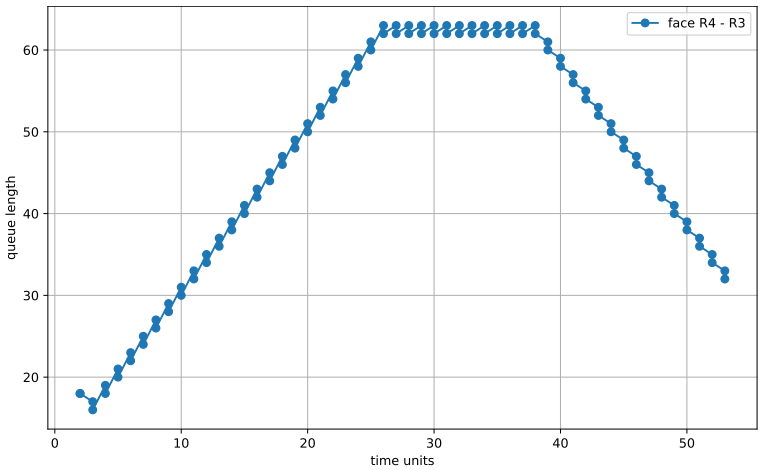


Figure 5.33 – Queue length at R_4 towards R_3 in expected rank-based forwarding approach in butterfly topology

[2]The generation size may be indicated in Interest/Data packets.

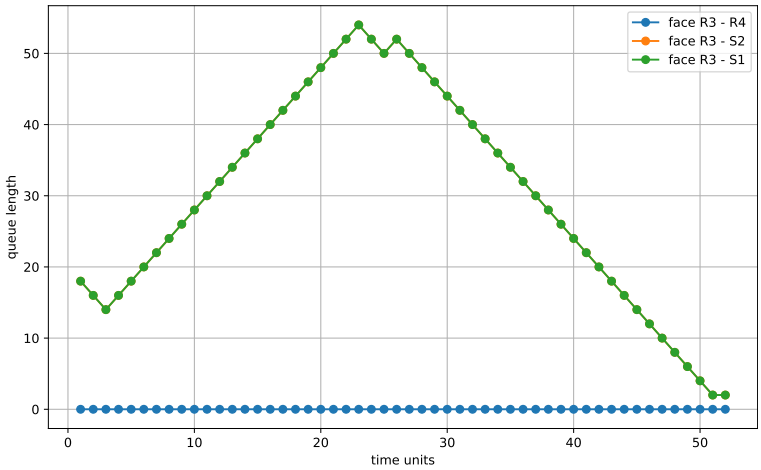


Figure 5.34 – Queue lengths at R_3 in expected rank-based forwarding approach in butterfly topology

Figure 5.33 presents the queue lengths at the node R_4 of the butterfly topology. In comparison to the queue lengths in Figure 5.4, observe that the nodes forward fewer Interests and stop forwarding when their expected rank estimate becomes equal to the generation size m.

Figure 5.34 presents the queue lengths at the node R_3 of the butterfly topology. Figure 5.34 shows the forwarding strategy adaptation at the node. At around time unit 25, we observe decreased queue lengths since the node stopped forwarding Interests. However, there is an increase at time 30 as node forwards Interest again due to the change in the value of $R_{e,k}$ as shown in Figure 5.35.

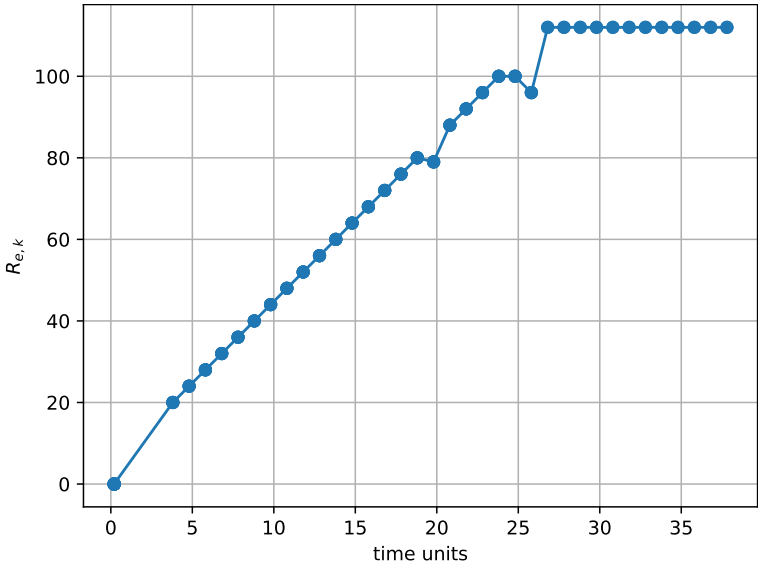
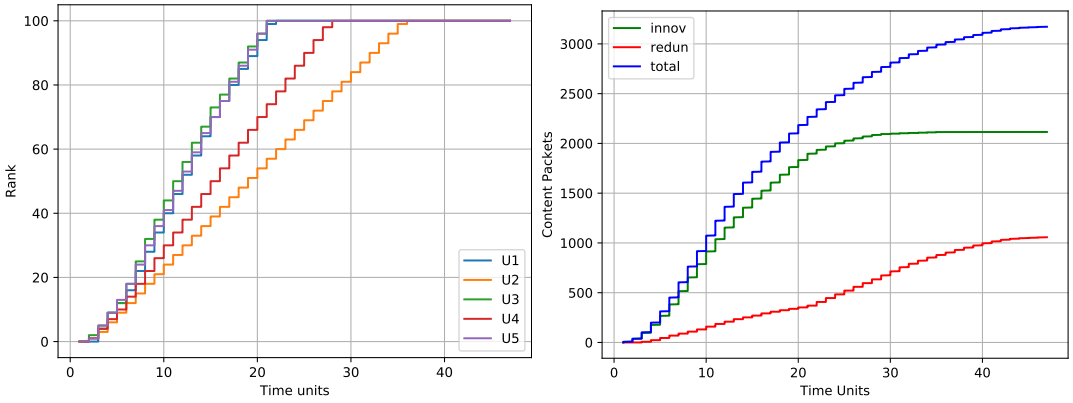


Figure 5.35 – Expected rank $R_{e,k}$ at nodeR_3 in expected rank-based forwarding approach in butterfly topology

Figures 5.36a and 5.36b presents the result with the expected rank-based strategy on the PlanetLab topology. Figure 5.37 shows that this strategy brings

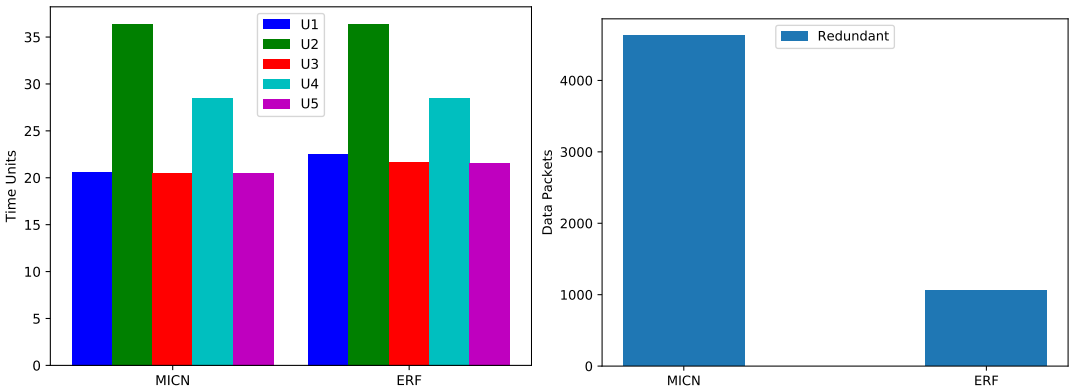
a good compromise in terms of the content retrieval time and the total data traffic.



(a) Rank evolution as a function of time

(b) Evolution of cumulative data traffic as a function of time

Figure 5.36 – Results of expected rank-based approach on PlanetLab topology



(a) Content retrieval time

(b) Redundant data traffic

Figure 5.37 – Results of expected rank-based forwarding (ERF) approach on PlanetLab topology

In comparison to the forwarding strategies presented in Sections 5.3.1-5.3.4, the expected rank-based forwarding strategy not only achieves the goal of maximum throughput but also effectively reduces the redundant traffic. This strategy provides the nodes the independence of choosing the forwarding based on the available information. No feedback or external tuning of decision parameters is required, the status of network is predicted using the computation of the expected rank (5.3) and the forwarding decision is taken based on the expected rank value.

5.4 Conclusion

MICN produces redundant traffic due to the lingering Interests that are a consequence of the flooding of Interests. The flooding of Interests is required to reach

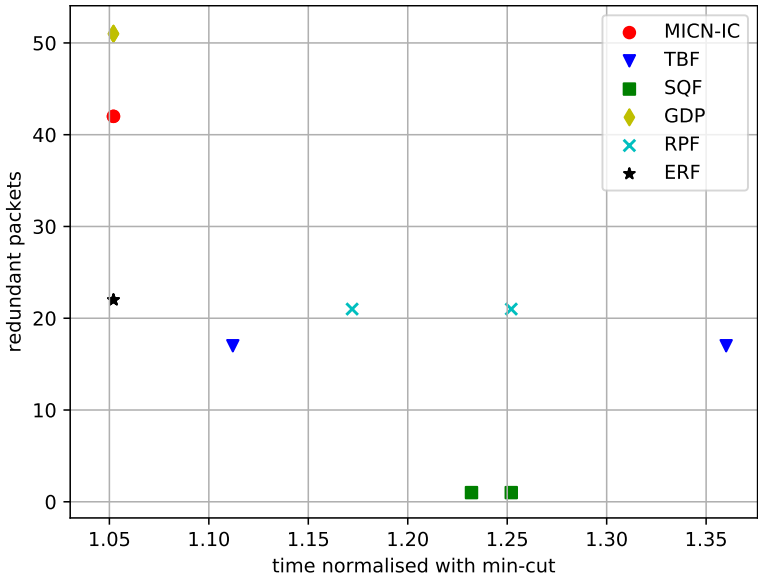


Figure 5.38 – Redundant traffic vs download time (two clients) in the Butterfly topology for different modified forwarding strategies

the network's capacity; however, the resulting redundant traffic deteriorates the performance in the case of consecutive generations retrieval. In this chapter, we propose modified forwarding strategies to be implemented at the nodes or clients to overcome lingering Interests in the network. The goal of these modified forwarding strategies is to reduce the redundant traffic while still ensuring fast download. Results from the proposed strategies provide insights into the forwarding strategies in MICN.

Figures 5.38 and 5.39 present the results obtained with the different heuristic-based strategies compared to Interest Cancellation (MICN-IC) presented in Section 4.5.2. Figure 5.38 presents the download time vs the amount of redundant data traffic in the Butterfly topology. Each forwarding strategy is presented

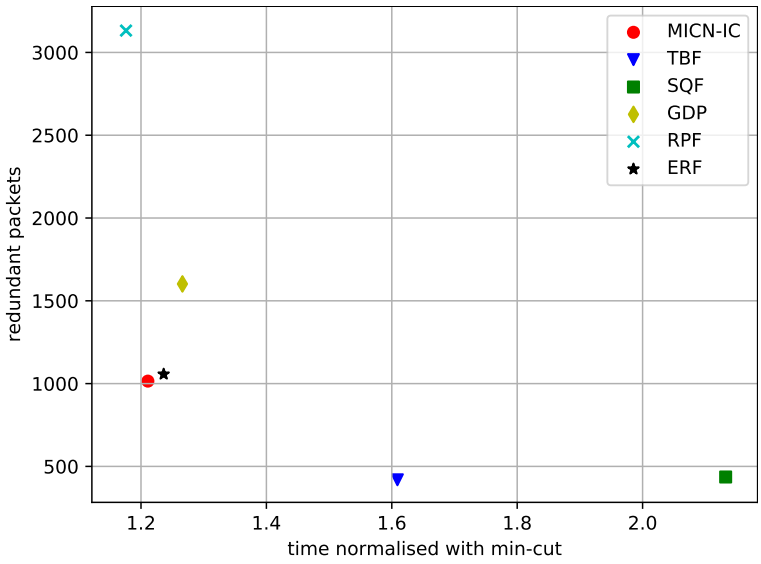


Figure 5.39 – Redundant traffic vs download time (average of 5 clients) in Butterfly topology for different modified forwarding strategies

with two points, each corresponding to one of the two clients in the Butterfly topology. Figure 5.39 presents the average download time[3] vs the redundant data traffic in the network in the PlanetLab topology. The results obtained with these strategies show that a tradeoff has to be found between the reduction of the the download time and the redundant traffic. The strategies that performed better with respect to time generated more traffic and alternatively the ones that reduced significantly the redundant traffic could not achieve even near maximum throughput.

In conclusion, the the heuristic-based forwarding strategies did help to reduce the redundant traffic in the network. Additionally, these strategies allowed us to identify the information available at the nodes and the parameters to limit the flooding in the network.

To conclude, the strategies helped us identify that although a multicast strategy helps to take advantage of network coding and receive content faster in MICN, the nodes may switch to different strategies. After the initial phase, the clients can choose a different forwarding strategy to reduce the network's lingering Interests and cumulative traffic. The findings from modified strategies open the way for future work to explore reinforcement learning techniques. The reinforcement learning can be used to implement adaptive forwarding strategy at the nodes for forwarding MILIC Interests.

[3]Average download time for the five clients in PlanetLab Topology is presented for the ease of presentation

6

Generalized Construction of Coding Sets

6.1 Introduction

In Chapter 4, we presented Multiple Interests for Linearly Independent Content (MILIC) as a solution to get back linearly independent content with each Interest sent in a network-coded NDN scenario.

The starting point for MILIC is to tackle the inefficiencies in network-coded NDN protocols that are mainly a consequence of classical NDN caching. For example, a node requesting coded content may receive an already received coded packet again from a neighboring cache. Assuming that a client targets high throughput, it will simultaneously send multiple parallel Interests as in Figure 6.1. One may be interested in a mechanism that guarantees that each of these Interests brings back innovative content.

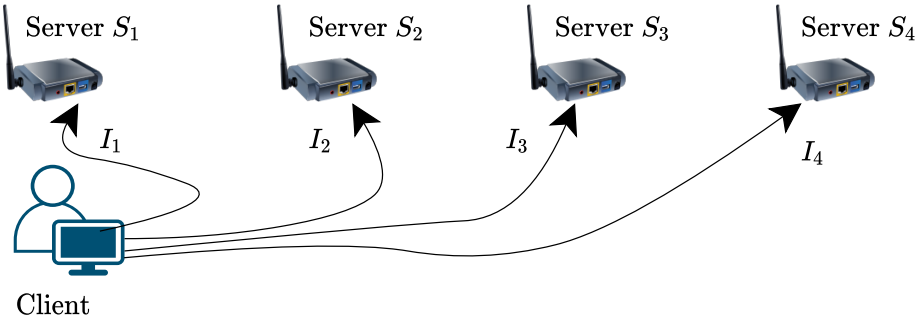


Figure 6.1 – Parallel transmission of multiple distinct Interests

The core idea of MILIC starts by supplementing each Interest with an index. MILIC defines indexed subsets such that each index returns content from the corresponding subset. These subsets are defined such that if one vector is chosen from each subset, the resulting set of vectors is linearly independent.

In this chapter, we introduce such subsets in a more general formal way as follows.

Problem 6.1 (The SELIT Problem)**.** Given a vector space of dimension n, is it possible to partition its elements such that the selected vectors are linearly independent when drawing arbitrarily one vector from each subset?

Solving the above Problem aims to find Sets Ensuring Linearly Independent Transversals (SELIT), where transversals means randomly drawing one vector from each subset (see Definition 6.2). MILIC provides a first solution to Problem 6.1 by defining the subsets $\mathcal{A}_k, k = 1, \ldots, n$ as in (4.1) (See Section 4.3.3). However, the size of the subsets is unbalanced and actually decreases with the increase of k (See Lemma 4.1). Thus there are more chances to be able to generate a coded segment from vectors in the first subsets than the last subsets, which is a problem for recoding (at the end of the generation retrieval). This chapter aims to find other SELIT constructions that potentially have sets of more balanced size.

The SELIT problem is loosely related to the well-known *index coding* problem [79] initially introduced in the satellite communication context by Birk *et al.* [79, 80]. Index coding is a canonical problem in network information theory that studies the fundamental limit and optimal coding schemes for broadcasting multiple messages to receivers with different side information.

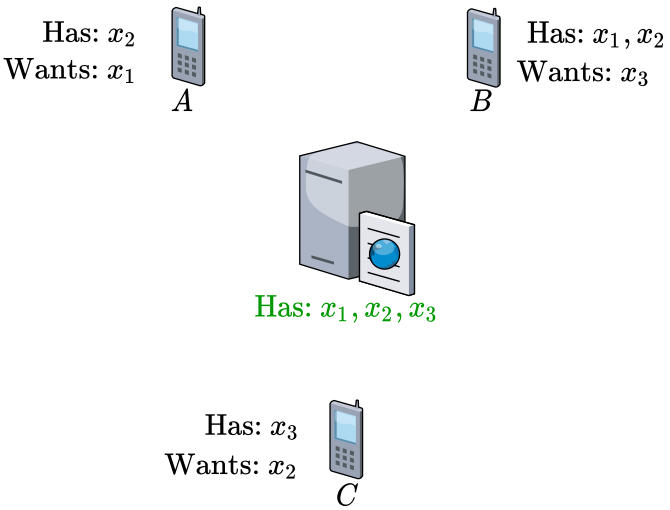


Figure 6.2 – Index coding problem: How can A, B, and C be satisfied with the minimum amount of transmissions ?

Consider, for example, a situation where a server has a wireless broadcast link to several clients. The clients already have some content and request more from the server, as shown in Figure 6.2. Index coding solutions determine the coding scheme to convey the clients' requested information with the minimum number of necessary transmissions.

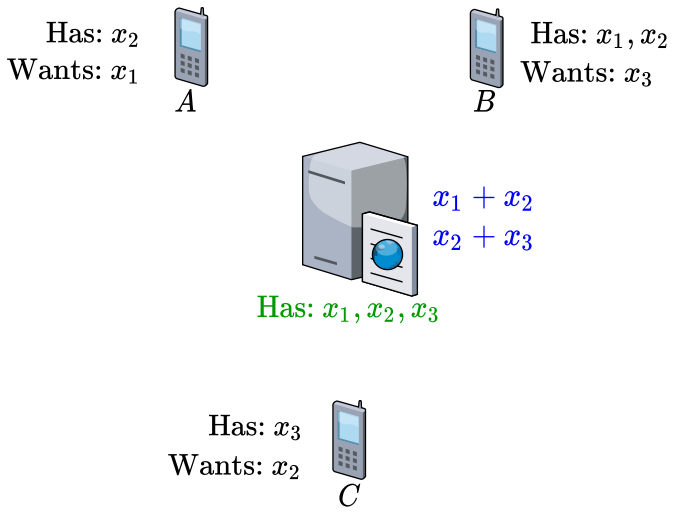


Figure 6.3 – Index coding problem: Source broadcasts coded messages (in blue) to satisfy A, B, and C with the minimum amount of transmissions

Example 6.1. Figure 6.2 presents a simple example of the index coding problem with a wireless communication system constituted of one source (in the middle) and three client nodes A, B, and C. The source stores three messages $x_1, x_2,$ and x_3. The client nodes are interested in subsets of these messages, with each one of them already having some information. The source has to find a way to convey all these messages to all the clients using the minimum possible number of broadcast transmissions.

A naive way would be to send all messages one by one for the clients, requiring three transmissions. Alternatively, as shown in Figure 6.3, the source can broadcast the coded messages $x_1 + x_3$ and $x_2 + x_3$. Each client node can then receive its desired message with only two transmissions.

The index coding research spawned results such as an equivalence with network coding [81] and capacity region results in the distributed case [82].

SELIT is simpler because it decouples the "capacity" aspect present in the index coding problem. SELIT addresses the issue of coding the source messages such that they are linearly independent. We are unaware of existing constructions of the literature that would provide solutions to the SELIT problem.

We study in detail some families of solutions for the SELIT problem. We will prove that a large class of generalizations of the MILIC construction are equivalent (*isomorphic*) to the MILIC construction (the family of subsets introduced in Chapter 4). The main result presented in this chapter is the proof for this fact in Theorem 6.4. This is of high interest because it limits the search space for alternate constructions, and the steps of the proof themselves shed light on properties the alternate solutions must have (or not have). Then, we exhibit an alternate family of solutions. Later we prove that a large family of possible solutions are equivalent to the MILIC construction or equivalent up to a permutation.

It is important to note here that this work is performed to investigate solutions other than MILIC. The dimension of the subsets in MILIC decrease, however, this does not have any affect on the performance of MICN due to re-encoding.

The chapter is organized as follows. We introduce the notations and formulate the SELIT problem is presented in Section 6.2. Section 6.3 discusses its solutions, including MILIC. In Section 6.4, a generic family of constructions (for potential solutions), is defined. Sections 6.5 and 6.6 contain the proof that solutions from this family are equivalent to the MILIC construction. Section 6.7 shows alternative constructions and Section 6.8 concludes.

6.2 Problem Formulation

In a network-coded NDN scenario, client nodes must send at least n Interests to retrieve a whole generation of n packets. With pure random network coding, we have to send more than n Interests per generation. The MILIC construction, defined in Section 4.3, guarantees that n Interests are sufficient to retrieve the content provided that each one of them reaches a source or content cache and is not lost. Additionally, if $n' < n$ Data packets are received, each of the n' coded segments are linearly independent with a high probability.

Motivated by these results, we determine whether alternate solutions can achieve similar results and if it is possible to have constructions with more balanced subset sizes. We looked for generalizations of our construction. We begin by presenting the mathematical formulation of the SELIT problem and then explore families similar to the MILIC construction that solve this problem.

6.2.1 Notations and Definitions

Let $[\![k]\!] = \{1, 2, \dots k\}$ be the set of integers from 1 to k. $\mathbb{F}$ denotes a finite field with $|\mathbb{F}| > 2$, $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$, and $\mathbb{F}^n$ is the vector space of dimension $n \geqslant 1$ over $\mathbb{F}$

$$\mathbb{F}^n = \left\{ \left(x_1, x_2, \dots, \quad x_n\right) \mid x_1 \in \mathbb{F}, x_2 \in \mathbb{F}, \dots, x_n \in \mathbb{F} \right\}.$$

The vector $e_i = (\underbrace{0, \dots, 0}_{i-1 \text{ zeros}}, 1, 0, \dots, 0)$ is the i-th canonical vector of $\mathbb{F}^n$.

Definition 6.1 (Encoding vector of a linear combination)**.** Consider a coded packet Q consisting of a linear combination of the source content $P_1, P_2, \dots P_n$, with $Q = \alpha_1 P_1 + \alpha_2 P_2 + \dots + \alpha_n P_n$, and $\alpha_i \in \mathbb{F}, \forall i \in [\![n]\!]$. The encoding vector of Q is $v = (\alpha_1, \alpha_2, \dots, \alpha_n)$.

Definition 6.2 (Transversal)**.** Let $\mathcal{E}$ be an arbitrary set. Consider a family $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_k)$ of k indexed subsets of $\mathcal{E}$. A *transversal* $\mathcal{T}$ of $\mathcal{A}$ is a tuple obtained by picking one element in each subset $\mathcal{A}_i$: $\mathcal{T} = (a_1, a_2, \dots, a_k)$ with $a_i \in \mathcal{A}_i, \forall i \in [\![k]\!]$.

Sets Ensuring Linearly Independent Transversals

Consider a client that wants to retrieve n segments $P_1, P_2, \dots, P_n$ of source content. The client sends n Interest packets $\{I_i\}_{i \in [\![n]\!]}$. The reply to the i-th Interest I_i from any server will be a linear combination whose encoding vector comes from a predefined set $\mathcal{A}_i$.

The problem considered in this chapter is to build a family of n sets $\{\mathcal{A}_i\}_{i\in[\![n]\!]}$ that satisfies the following properties.

- **Completeness:** When n coded packets with encoding vectors $v_1 \in \mathcal{A}_1, v_2 \in \mathcal{A}_2, \ldots, v_n \in \mathcal{A}_n$ are received by a client, this client should be able to recover $P_1, P_2, \ldots, P_n$.

- **High diversity:** If k coded packets with encoding vectors $v_1, \ldots, v_k$ are received, some potentially drawn from the same sets, the receiving node should be able to retrieve k linearly independent coded segments from the coded packets with high probability.

Example 6.2. Consider a node receiving four segments $v_1 \in \mathcal{A}_7, v_2 \in \mathcal{A}_7, v_3 \in \mathcal{A}_5, v_4 \in \mathcal{A}_5$. They should provide non-redundant information, *i.e.,* the node should have four linearly independent packets with high probability. This can also be stated as, for k packets, the number of linearly independent coded segments of the receiving node should be k with high probability.

The MILIC construction (see Definition 6.3 below), introduced in Chapter 4, satisfies both properties. Completeness is satisfied by construction, and high diversity is proven by Properties 4.2 and 4.3 (see Section 4.3.2). Since high diversity is arguably more difficult to formulate precisely for being inherently probabilistic, we ignore this aspect[1]. Instead, we focus on ensuring the completeness property. This can be recast as finding Sets Ensuring Linearly Independent Transversals (SELIT) defined as follows

SELIT Problem of dimension n Given a finite field $\mathbb{F}$, and the vector space $\mathbb{F}^n$ of dimension n over $\mathbb{F}$: Find $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$, a family of n disjoint subsets of $\mathbb{F}$, such that any of their transversals (see Definition 6.2) always constitutes a set of linearly independent vectors.

Clearly, a family $\mathcal{A}$ of sets solution of the SELIT problem satisfy the completeness property. If encoding vectors are not linearly independent, it is not possible to recover n source packets.

6.3 SELIT Solutions

To illustrate the SELIT problem, we present an example of SELIT solution for $n = 2$.

Example 6.3.

Consider $\mathbb{F} = \mathrm{GF}(3), n = 2, k = 2, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$[2] with

$\mathcal{A}_1 = \{(1,2), (2,1)\}$,
$\mathcal{A}_2 = \{(1,0), (1,1)\}$.

There are exactly 4 possible transversals (obtained by enumeration):

[1] Notice that high diversity is correlated with the high cardinality of the sets $\mathcal{A}_i$ in any case. So after finding large $\mathcal{A}_i$s, one may check *a posteriori* their diversity.

[2] Note that $\mathcal{A}$ here is not a partition of $\mathbb{F}^n$ or $\mathbb{F}^n/\{0\}$

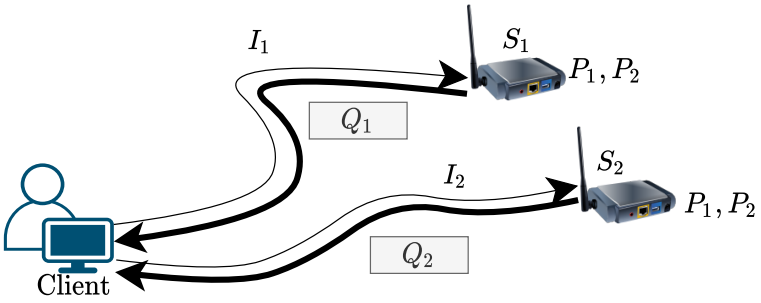


Figure 6.4 – Simple SELIT example

- $\mathcal{T}_1 = \big((1,2),(1,0)\big)$
- $\mathcal{T}_2 = \big((1,2),(1,1)\big)$
- $\mathcal{T}_3 = \big((2,1),(1,0)\big)$
- $\mathcal{T}_4 = \big((2,1),(1,1)\big)$

We can compute the determinant of each of the encoding vectors corresponding to the transversals to show that they are linearly independent.

- $\det(\mathcal{T}_1) = \left|\begin{smallmatrix}1&2\\1&0\end{smallmatrix}\right| = 1 \neq 0$
- $\det(\mathcal{T}_2) = \left|\begin{smallmatrix}1&2\\1&1\end{smallmatrix}\right| = 2 \neq 0$
- $\det(\mathcal{T}_3) = \left|\begin{smallmatrix}2&1\\1&0\end{smallmatrix}\right| = 1 \neq 0$
- $\det(\mathcal{T}_4) = \left|\begin{smallmatrix}2&1\\1&1\end{smallmatrix}\right| = 1 \neq 0$

Hence all 4 transversals are sets of 2 linearly independent vectors, and thus $\mathcal{A}$ is a SELIT solution.

Consider two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ defined as in Example 6.3. This can be seen as the problem presented in Figure 6.4 where clients send indexed Interests I_1 and I_2 to retrieve a content with two source segments P_1 and P_2.

The indexed Interests I_1 and I_2 bring back contents belonging to their respective subsets $\mathcal{A}_1$ and $\mathcal{A}_2$. A possible reply to Interest I_1 is either $Q_1 = P_1 + 2P_2$ or $Q_1' = 2P_1 + P_2$. These replies are linearly independent of any of the two possible replies to I_2, which are $Q_2 = P_1$ and $Q_2' = P_1 + P_2$.

Definition 6.3 (Multiple Interests for Linearly Independent Contents)**.** MILIC is a construction with sets of encoding vectors $\{\mathcal{A}_i\}_{i\in[\![n]\!]}$ introduced in Section 4.3 as follows: $\forall i \in [\![n]\!]$,

$$\mathcal{A}_i = \{(v_1, ..., v_n) \in \mathbb{F}^n \mid v_i \neq 0 \text{ and } \forall j \in [\![i-1]\!], v_j = 0\}.$$

Example 6.4 (MILIC for $n = 3$)**.**

- $\mathcal{A}_1 = \big\{\big(a_1,\ \ a_2,\ \ a_3\big) \mid a_1 \in \mathbb{F}^*, a_2 \in \mathbb{F}, a_3 \in \mathbb{F}\big\}$

- $\mathcal{A}_2 = \{(0, \quad b_2, \quad b_3) \mid b_2 \in \mathbb{F}^*, b_3 \in \mathbb{F}\}$
- $\mathcal{A}_3 = \{(0, \quad 0, \quad c_3) \mid c_3 \in \mathbb{F}^*\}$

Example 6.4 illustrates the MILIC construction introduced in Section 4.3.3 for $n = k = 3$. MILIC is one possible solution to the SELIT problem. By construction, any transversal of the MILIC sets is linearly independent. A triangular matrix is obtained when taking row vectors from each of the subsets of a MILIC construction.

Later in this chapter, we attempt to generalize MILIC constructions with matrices. All elements of these matrices are freely picked from predefined sets for any field $\mathbb{F}$ with $|\mathbb{F}| > 2$ (*i.e.,* all except $GF(2)$) with additional constraints, see Definition 6.6. Theorem 6.4 states that generalizations of the MILIC construction are MILIC constructions up to a permutation of indices.

6.4 Families of SELIT Constructions

Motivated by the results of MILIC as a SELIT solution, we investigate other families of SELIT solutions. One naïve solution is to have a family of subsets containing one vector only or a family with only one non-zero element at one position. Nevertheless, performing network coding with such families is meaningless since a coded segment is equivalent to a source segment. Hence such families are not interesting for network coding.

To reduce size of the search space for possible SELIT solutions a special family of sets that are of the same form as that of MILIC are explored. This section specifies the notion of similarity, after introducing some additional definitions.

6.4.1 Canonical Family of Sets

If $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a SELIT solution, one can take an arbitrary vector $v_i \in \mathcal{A}_i$ in each set, and use the set of vectors $\{v_i\}_{i \in [\![n]\!]}$ as a new basis coordinate. Accordingly, we introduce the concept of canonical family of sets:

Definition 6.4 (Canonical family of sets)**.** Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a family of n subsets of $\mathbb{F}^n$. A *canonical* family of sets is such that $e_i \in \mathcal{A}_i$ for all $i \in [\![n]\!]$.

6.4.2 Canonical and Component-Wise Family of Sets

Definition 6.5 (Canonical and component-wise family of sets)**.** Consider n subsets of $\mathbb{F}, \mathcal{A}_{i,j} \in \mathbb{F}, i \in [\![n]\!], j \in [\![n]\!]$. A family $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ of n subsets of $\mathbb{F}^n$ is a canonical and component-wise family of sets if the subset $\mathcal{A}_i, i \in [\![n]\!]$ is the union of e_i with the n-fold Cartesian product of sets $\mathcal{A}_{i,j}, j \in [\![n]\!]$, *i.e.,*

$$\mathcal{A}_i = \{(v_1, v_2, \ldots, v_n) \mid v_j \in \mathcal{A}_{i,j} \ \forall j \in [\![n]\!]\} \cup \{e_i\}. \tag{6.1}$$

Note that each element can be changed independently, which is an important property. A canonical and component-wise family of sets $\mathcal{A}$ is fully specified by

the table $\mathcal{C}(\mathcal{A})$ of subsets $\mathcal{A}_{i,j}$ denoted as

$$\mathcal{C}(\mathcal{A}) \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{A}_{1,1} & \cdots & \mathcal{A}_{1,n} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{n,1} & \cdots & \mathcal{A}_{n,n} \end{bmatrix}. \tag{6.2}$$

Remark 6.1. As defined in Section 6.2.1, only field size $|\mathbb{F}| > 2$ are considered, since in the case of $\mathbb{F} = 2$ there are not many choices of coefficients and the proofs may not apply.

Example 6.5. The MILIC construction $\mathcal{A}^{\text{MILIC}}$ in Definition 6.3 is a canonical and component-wise family, associated with the subsets $\mathcal{A}_{i,j}^{\text{MILIC}}$ defined as follows:

$$\mathcal{A}_{i,j}^{\text{MILIC}} = \begin{cases} \mathbb{F}^* & \text{if } i = j \\ \{0\} & \text{if } i < j \\ \mathbb{F} & \text{if } i > j \end{cases}$$

and we have

$$\mathcal{C}(\mathcal{A}^{\text{MILIC}}) = \begin{bmatrix} \mathbb{F}^* & \mathbb{F} & \cdots & \mathbb{F} \\ \{0\} & \mathbb{F}^* & \cdots & \mathbb{F} \\ \vdots & \vdots & \ddots & \mathbb{F} \\ \{0\} & \{0\} & \cdots & \mathbb{F}^* \end{bmatrix}.$$

The Cartesian products defined in the MILIC construction already include the canonical unit vectors e_i.

Consider $\mathbb{F} = GF(3)$, and the MILIC construction $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ with $n = 3$; $\mathbb{F}$ can then be represented as $\mathbb{F} = \{0, 1, 2\}$. Then

$$\mathcal{C}(\mathcal{A}) = \begin{bmatrix} \{1,2\} & \{0,1,2\} & \{0,1,2\} \\ \{0\} & \{1,2\} & \{0,1,2\} \\ \{0\} & \{0\} & \{1,2\} \end{bmatrix}$$

6.4.3 Canonical, Diagonal and Restricted Diagonal Family of Sets

An important property of the MILIC construction represented $\mathcal{C}(\mathcal{A}^{\text{MILIC}})$ is that $\mathcal{C}(\mathcal{A}^{\text{MILIC}})$ has non-zero diagonals. In addition to the canonical and component-wise families of sets we can define diagonal and restricted diagonal families of sets.

Definition 6.6 (Canonical, Diagonal and Restricted Diagonal family of sets)**.** Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a canonical and component-wise family of n sets of $\mathbb{F}^n$. A diagonal family is component-wise canonical, if the sets $\mathcal{A}_{i,i}$ satisfy $\mathcal{A}_{i,i} = \mathbb{F}^*$, *i.e.,*

$$\mathcal{C}(\mathcal{A}) = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{1,2} & \cdots & \mathcal{A}_{1,n} \\ \mathcal{A}_{2,1} & \mathbb{F}^* & \cdots & \mathcal{A}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}_{n,1} & \mathcal{A}_{n,2} & \cdots & \mathbb{F}^* \end{bmatrix}. \tag{6.3}$$

A canonical family is a *restricted diagonal* family if the off-diagonal sets $\mathcal{A}_{i,j}$ of $\mathcal{C}(\mathcal{A})$ additionally satisfy:

$$\forall i \in [\![n]\!], \forall j \in [\![n]\!] \setminus \{i\} : \mathcal{A}_{i,j} \subset \mathbb{F}^* \text{ or } \mathcal{A}_{i,j} = \{0\}. \tag{6.4}$$

Note that a restricted diagonal family of sets is also a diagonal family of sets.

6.5 Deriving SELIT Solutions of Smaller Dimension

One important property of a canonical component-wise SELIT solution $\mathcal{A}$ for a given n is that one can derive SELIT solutions of smaller dimension $\mathcal{A}'$ for $n' < n$ by removing some sets and some rows or columns of the sets. This is shown in Theorem 6.1 at the end of this section and will be used as the basis of the proof by induction of our main result in Section 6.6.

We first formally define what is meant by *derivation*, through the definition of a *truncated permutation* which consists in applying some permutation π to the elements of a vector, then truncating it to the first k elements.

Definition 1. A *k-truncated permutation* π of $[\![n]\!]$ is an injective function $\pi : [\![k]\!] \longrightarrow [\![n]\!]$, *i.e.*, $\pi_i \neq \pi_j$ when $i, j \in [\![k]\!]$, $i \neq j$. The indices of the coefficients are specified by a tuple of k different indices $L = (\pi_1, \pi_2, \cdots, \pi_k)$. The term *mapping* denotes the application of a truncated permutation to the elements of the vector.

Definition 6.7 (Mapping of a vector)**.** Let $v = \left(v_1, v_2, \cdots, v_n\right) \in \mathbb{F}^n$ be a vector. The mapping π (k-truncated permutation) of $[\![n]\!]$ is

$$\mathrm{TP}\left(v, \pi\right) = \left(v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_k}\right) \in \mathbb{F}^k. \tag{6.5}$$

This definition can be extended to a family of subsets in a way that their diagonal structure is preserved.

Definition 6.8 (Mapping of a family of sets)**.** Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a family of subsets of $\mathbb{F}^n$. The *mapping* of $\mathcal{A}$, denoted $\mathrm{TP}\left(\mathcal{A}, \pi\right)$ is the family $\mathcal{A}'$ of subsets of $\mathbb{F}^n$ such that

$$\begin{aligned} \mathcal{A}' &= (\mathcal{A}'_1, \ldots, \mathcal{A}'_k) \\ \mathcal{A}'_i &= \left\{\mathrm{TP}(v, \pi_1, \pi_2, \ldots, \pi_k) \,|\, \forall v \in \mathcal{A}_{\pi_i}\right\}, \forall i \in [\![k]\!]. \end{aligned} \tag{6.6}$$

If $\mathcal{A}$ is a component-wise family with coefficients from the sets $(\mathcal{A}_{ij})_{i \in [\![n]\!], j \in [\![n]\!]}$, then we introduce a notation for the table of coefficients of the mapping of $\mathcal{A}$:

$$\mathcal{P}\left(\mathcal{A}, \pi\right) \stackrel{\text{def}}{=} \mathcal{C}\left(\mathrm{TP}\left(\mathcal{A}, \pi\right)\right)$$

$$\mathcal{P}\left(\mathcal{A}, \pi\right) = \begin{bmatrix} \mathcal{A}_{\pi_1\pi_1} & \mathcal{A}_{\pi_1\pi_2} & \cdots & \mathcal{A}_{\pi_1\pi_k} \\ \mathcal{A}_{\pi_2\pi_1} & \mathcal{A}_{\pi_2\pi_2} & \cdots & \mathcal{A}_{\pi_2\pi_k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}_{\pi_k\pi_1} & \mathcal{A}_{\pi_n\pi_2} & \cdots & \mathcal{A}_{\pi_k\pi_k} \end{bmatrix}. \tag{6.7}$$

Example 6.6. For the MILIC construction with $n = 3$ (introduced in Example 6.5), we have the following examples of mapping:

$$\mathcal{P}(\mathcal{A}, (1,2,3)) = \begin{bmatrix} \mathbb{F}^* & \mathbb{F} & \mathbb{F} \\ \{0\} & \mathbb{F}^* & \mathbb{F} \\ \{0\} & \{0\} & \mathbb{F}^* \end{bmatrix} \text{ with } \begin{matrix} \mathbb{F} = \{0,1,2\} \\ \mathbb{F}^* = \{1,2\} \end{matrix}$$

$$\mathcal{P}(\mathcal{A}, (1,2)) = \begin{bmatrix} \mathbb{F}^* & \mathbb{F} \\ \{0\} & \mathbb{F}^* \end{bmatrix} \text{ and } \mathcal{P}(\mathcal{A}, (2,1)) = \begin{bmatrix} \mathbb{F}^* & \{0\} \\ \mathbb{F} & \mathbb{F}^* \end{bmatrix}.$$

When $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ is a MILIC construction, then for any set of k indices $\pi_1, \pi_2, \ldots, \pi_k$ with $\pi_1 < \pi_2 < \ldots < \pi_k$, we have:

$$\mathcal{P}(\mathcal{A}, (\pi_1, \pi_2, \ldots, \pi_k)) = \begin{bmatrix} \mathbb{F}^* & \mathbb{F} & \mathbb{F} & \cdots & \mathbb{F} \\ \{0\} & \mathbb{F}^* & \mathbb{F} & \cdots & \mathbb{F} \\ \{0\} & \{0\} & \mathbb{F}^* & \cdots & \mathbb{F} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \{0\} & \{0\} & \{0\} & \cdots & \mathbb{F}^* \end{bmatrix} \tag{6.8}$$

Theorem 6.1 (Mapping Theorem for Canonical Solutions)**.** Let $\mathcal{A}$ be a canonical family of sets, which is a solution to the SELIT problem if dimension n, and π a k -truncated permutation of $\mathcal{A}$.

Then $\mathcal{A}' = \mathrm{TP}(\mathcal{A}, \pi)$ is a family of k subsets of $\mathbb{F}^k$ and a solution to the SELIT problem of dimension k.

Proof. By contradiction: assume $\mathcal{A}'$ is not a SELIT solution, this implies that there exist $w_i \in \mathcal{A}'_i, \forall i \in [\![k]\!]$ and $\alpha \in \mathbb{F}^k$, $\alpha \neq 0$ such that $\alpha_1 w_1 + \alpha_2 w_2 + \ldots + \alpha_k w_k = 0$. $\mathcal{A}'$ is a family of subsets that are mappings of $\mathcal{A}$, which implies from (6.7) that each vector $w_i \in \mathcal{A}'_i$ is a mapping of some vector $v_i \in \mathcal{A}_i$. Then

$$\sum_{i \in [\![k]\!]} \alpha_i v_i = \big(\underbrace{0, 0, \cdots, 0}_{k \text{ zeros}}, x_{k+1}, x_{k+2}, \cdots, x_n\big)$$

up to a permutation of the elements. Since $\mathcal{A}$ is a canonical family of sets, each $\mathcal{A}_j$ includes the canonical vector e_j, and then

$$\sum_{i=1}^{k} \alpha_i v_i - \sum_{i=k+1}^{n} x_i e_i = 0$$

which contradicts the fact that $\mathcal{A}$ is a SELIT solution. Hence $\mathcal{A}'$ is a SELIT solution. □

6.6 Properties of Canonical and Restricted Diagonal Solutions of the SELIT Problem

In this section, we consider one solution of the SELIT problem $\mathcal{A}$ that is canonical, component-wise, and restricted diagonal. From the previous definitions we

can write

$$\begin{aligned}\mathcal{A}_i &= \{\begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \mid v_j \in \mathcal{A}_{i,j}\ \forall j = 1 \dots n\} \cup \{e_i\} \\ \mathcal{C}(\mathcal{A}) &= \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{1,2} & \cdots & \mathcal{A}_{1,n} \\ \mathcal{A}_{2,1} & \mathbb{F}^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathcal{A}_{n-1,n} \\ \mathcal{A}_{n,1} & \cdots & \mathcal{A}_{n,n-1} & \mathbb{F}^* \end{bmatrix} \\ &\text{with } \mathcal{A}_{i,j} \subset \mathbb{F}^* \text{ or } \mathcal{A}_{i,j} = \{0\} : \forall i \in [\![n]\!], \forall j \in [\![n]\!] \setminus \{i\}.\end{aligned} \tag{6.9}$$

Consider the 2×2 mapping,

$$\mathcal{P}(\mathcal{A}, \pi) = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{\pi_1,\pi_2} \\ \mathcal{A}_{\pi_2,\pi_1} & \mathbb{F}^* \end{bmatrix}$$

involves only $\mathcal{A}_{\pi_1,\pi_2}$ and $\mathcal{A}_{\pi_2,\pi_1}$, that must satisfy the property introduced in Lemma 6.1:

Lemma 6.1 (Non-Zero Set Exclusion)**.** Consider $n > 1$. Let $\mathcal{A}$ be the SELIT solution introduced in (6.9). Consider the 2×2 mapping of $\mathcal{A}$.

Then $\mathcal{A}_{\pi_1,\pi_2} = \{0\}$ or $\mathcal{A}_{\pi_2,\pi_1} = \{0\}$ or $\mathcal{A}_{\pi_1,\pi_2} = \mathcal{A}_{\pi_2,\pi_1} = \{0\}$.

Proof. By contradiction: Assume that $\mathcal{A}_{\pi_1,\pi_2} \neq \{0\}$ and $\mathcal{A}_{\pi_2,\pi_1} \neq \{0\}$. Then from (6.9), $\mathcal{A}_{\pi_1,\pi_2} \subset \mathbb{F}^*$, and $\mathcal{A}_{\pi_2,\pi_1} \subset \mathbb{F}^*$, hence $0 \notin \mathcal{A}_{\pi_1,\pi_2}$ and $0 \notin \mathcal{A}_{\pi_2,\pi_1}$. Let $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ be the mapping of $\mathcal{A}$, *i.e.* $\mathcal{A}' = TP(\mathcal{A}, \pi)$. We know that:

$$\mathcal{C}(\mathcal{A}') = \mathcal{P}(\mathcal{A}, \pi) = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{\pi_1,\pi_2} \\ \mathcal{A}_{\pi_2,\pi_1} & \mathbb{F}^* \end{bmatrix}.$$

Take $v_1 = \begin{pmatrix} a, a_1 \end{pmatrix}$ and $v_2 = \begin{pmatrix} a_2, a' \end{pmatrix}$ one has $a_1 = 0$ and $a_2 = 0$. Since $a \in \mathbb{F}^*$ and $a' \in \mathbb{F}^*$ they can be chosen such that $a = a_1$ and $a' = a_2$. Then

$$\det(v_1, v_2) = \begin{vmatrix} a_1 & a_1 \\ a_2 & a_2 \end{vmatrix} = 0,$$

so v_1, v_2 are linearly dependent.

Thus $\mathcal{A}'$ is not a SELIT solution and by Theorem 6.1, neither is $\mathcal{A}$. This is a contradiction, therefore the initial assumption that both $\mathcal{A}_{\pi_1,\pi_2}$ and $\mathcal{A}_{\pi_2,\pi_1}$ are not $\{0\}$ must be false, hence the lemma. □

This lemma allows us to further characterize SELIT solutions using binary relations. The binary relation give us an order that ultimately provides us the permutation that is required to get the MILIC matrix.

Definition 6.9 (Binary Relations from $\mathcal{A}$)**.** Consider $\mathcal{A}$ as defined by (6.9), and a k-truncated permutation π of $[\![n]\!]$. We can introduce a binary relation on π_1 and π_2, depending on whether $\mathcal{A}_{\pi_1,\pi_2}$ and $\mathcal{A}_{\pi_2,\pi_1}$ are $\{0\}$:

$$\pi_1 \overset{\mathcal{A}}{\prec} \pi_2 \text{ if } \mathcal{A}_{\pi_2,\pi_1} = \{0\} \text{ and } \mathcal{A}_{\pi_1,\pi_2} \neq \{0\}; \text{thus } \mathcal{P}(\mathcal{A}, \pi) = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{\pi_1,\pi_2} \\ \{0\} & \mathbb{F}^* \end{bmatrix}, \tag{6.10}$$

$$\pi_2 \overset{\mathcal{A}}{\prec} \pi_1 \text{ if } \mathcal{A}_{\pi_1,\pi_2} = \{0\} \text{ and } \mathcal{A}_{\pi_2,\pi_1} \neq \{0\}; \text{thus } \mathcal{P}(\mathcal{A}, \pi) = \begin{bmatrix} \mathbb{F}^* & \{0\} \\ \mathcal{A}_{\pi_2,\pi_1} & \mathbb{F}^* \end{bmatrix}, \tag{6.11}$$

$$\pi_2 \overset{\mathcal{A}}{\sim} \pi_1 \text{ if } \mathcal{A}_{\pi_1,\pi_2} = \mathcal{A}_{\pi_2,\pi_1} = \{0\}; \text{thus } \mathcal{P}(\mathcal{A}, \pi) = \begin{bmatrix} \mathbb{F}^* & \{0\} \\ \{0\} & \mathbb{F}^* \end{bmatrix}. \tag{6.12}$$

Observe that

$$\pi_1 \overset{\mathcal{A}}{\not\prec} \pi_2 \iff \mathcal{A}_{\pi_1,\pi_2} = \{0\} \tag{6.13}$$

The over-script $\mathcal{A}$ makes clear that the binary relations depend on $\mathcal{A}$. To simplify, when there is no ambiguity from the context, we will write $\prec$ instead of $\overset{\mathcal{A}}{\prec}$.

We now start with a lemma on the determinant of matrices.

Lemma 6.2. Consider the sets $\mathcal{B}_{i,j}$, $j \in [\![k]\!], i \in [\![j-1]\!]$, the sets $\mathcal{D}_i \subset \mathbb{F}^*$, $i \in [\![k]\!]$, and the set of matrices $\mathcal{M}_k$ defined as

$$\mathcal{M}_k = \left\{ \begin{bmatrix} d_1 & b_{1,2} & b_{1,3} & \cdots & b_{1,k-1} & b_{1,k} \\ c_1 & d_2 & b_{2,3} & \cdots & b_{2,k-1} & b_{2,k} \\ 0 & c_2 & d_3 & \cdots & b_{3,k-1} & b_{3,k} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & d_{k-1} & b_{k-1,k} \\ 0 & 0 & 0 & \cdots & c_{k-1} & d_k \end{bmatrix} \right.$$

$$\text{where } b_{i,j} \in \mathcal{B}_{i,j}, \forall j \in [\![k]\!], \forall i \in [\![j-1]\!],$$

$$\left. \text{and } d_i \in \mathcal{D}_i, \forall i \in [\![k]\!], \text{ and } c_i \in \mathbb{F}^*, \forall i \in [\![k-1]\!], \right.$$

then there exists a matrix $M \in \mathcal{M}_k$ such that $\det(M) \neq 0$.

Proof. The proof is by induction. For $k = 2$, consider a 2×2 matrix $M_2 \in \mathcal{M}$

$$M_2 = \begin{bmatrix} d_1 & b_{1,2} \\ c_1 & d_2 \end{bmatrix}$$

where d_1 and d_2, arbitrarily chosen from the sets $\mathcal{D}_i \subset \mathbb{F}^*$, $i = 1, 2$, are necessarily non-zero; $b_{1,2} \in \mathcal{B}_{1,2}$ is also arbitrary chosen, and might be zero. If $b_{1,2} = 0$ then $\det(M_2) \neq 0$. Otherwise if $b_{1,2} \neq 0$ then since $c \in \mathbb{F}^*$, there exists c_1 such that $\det(M_2) \neq 0$. Thus the result for $k = 2$.

Assume the property is satisfied for $\mathcal{M}_{k-1}$ for some $k > 1$. The determinant of a $k \times k$ matrix $M_k \in \mathcal{M}_k$ is

$$\det(M_k) = d_1 \det(M'_{k-1}) - c_1 \det(Q)$$

where $M'_{k-1} \in \mathcal{M}_{k-1}$ and Q is some $(k-1) \times (k-1)$ matrix. Neither M'_{k-1} nor Q involve the coefficient c_1. From the induction assumption, we know that there exists $M'_{k-1} \in \mathcal{M}_{k-1}$ such that $\det(M'_{k-1}) \neq 0$. The matrix M_k is built from M'_{k-1}, selecting arbitrarily the remaining elements from their possible sets, except c_1. If $\det(Q) = 0$, then $\forall c_1 \in \mathbb{F}^*$, $\det(M_k) \neq 0$. If $\det(Q) \neq 0$, $\det(M_k)$ will take as many different values as c_1 and $\exists\ c_1 \in \mathbb{F}^*$ such that $\det(M_k) \neq 0$. Since the property is true for $\mathcal{M}_2$, the lemma is proven. □

Theorem 6.2. Let $\mathcal{A}$ be a SELIT solution as in (6.9). Let $k > 1$. Consider a sequence of k indices $\mathrm{L} = \{\pi_1, \pi_2, \ldots, \pi_k\}$, such that $\pi_1 \prec \pi_2, \pi_2 \prec \pi_3, \ldots, \pi_{k-1} \prec \pi_k$; then $\pi_k \not\prec \pi_1$.

Proof. We prove this theorem by induction. From property in (6.10) and (6.11), $\pi_1 \prec \pi_2$ and $\pi_2 \prec \pi_1$ are mutually exclusive, and the theorem is proven for $k = 2$. For $k > 2$, by induction, assuming that it was satisfied for $2, 3, \ldots, k-1$.

Consider a sequence of k indices $\mathrm{L} = \{\pi_1, \pi_2, \ldots, \pi_k\}$, that satisfies

$$\pi_1 \prec \pi_2, \pi_2 \prec \pi_3, \ldots, \pi_{k-1} \prec \pi_k.$$

For any $i \in [\![k]\!]$ and $j \in [\![k]\!]$ with $i < j$ and $(i,j) \neq (1,k)$, we can apply the induction hypothesis for $\pi_i \prec \pi_{i+1}, \pi_{i+1} \prec \pi_{i+2}, \ldots,$ and $\pi_{j-1} \prec \pi_j$, which forms a chain of $k' = j - i + 1 \leq k - 1$ binary relations. Thus $\pi_j \not\prec \pi_i$ and from (6.13),

$$\forall i \in [\![k]\!], \forall j \in [\![k]\!] \text{ with } i < j, (i,j) \neq (1,k) : \mathcal{A}_{\pi_j, \pi_i} = \{0\}.$$

Notice that these are all the elements in the lower triangle of the table $\mathcal{P}(\mathcal{A}, \pi)$, and are $\{0\}$, except for the diagonal and for $\mathcal{A}_{\pi_k, \pi_1}$.

$$P_{\mathrm{L}} = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{\pi_1,\pi_2} & \cdots & \mathcal{A}_{\pi_1,\pi_{k-1}} & \mathcal{A}_{\pi_1,\pi_k} \\ \{0\} & \mathbb{F}^* & \cdots & \mathcal{A}_{\pi_2,\pi_{k-1}} & \mathcal{A}_{\pi_2,\pi_k} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \{0\} & \{0\} & \ddots & \mathbb{F}^* & \mathcal{A}_{\pi_{k-1},\pi_k} \\ \mathcal{A}_{\pi_k,\pi_1} & \{0\} & \cdots & \{0\} & \mathbb{F}^* \end{bmatrix} \tag{6.14}$$

where

$$\begin{cases} \mathcal{A}_{i,i} = \mathbb{F}^* & \text{if } i = j, \\ \mathcal{A}_{i,j} \subset \mathbb{F}^* & \text{if } j = i + 1, \\ \mathcal{A}_{i,j} \subset \mathbb{F} & \text{if } j > i + 1, \\ \mathcal{A}_{i,j} = 0 & \text{if } i < j. \end{cases}$$

Consider $\mathcal{A}' = \mathrm{TP}(\mathcal{A}, \pi)$, a mapping of the SELIT solution $\mathcal{A}$. Consider the set of matrices $\mathcal{Q}_{\mathcal{A}'}$ constructed from vectors obtained by picking one vector in each set of $\mathcal{A}'$. Each matrix $M = (m_{i,j})$ in $\mathcal{Q}_{\mathcal{A}'}$ is obtained by picking a coefficient in the table P_{L} in (6.14) at the corresponding position. Its determinant can be written as

$$\det(M) = m_{1,1} \det(G) - m_{k,1} \det(H)$$

where $m_{1,1} \in \mathbb{F}^*$, G is a upper triangular matrix (with diagonal elements in $\mathbb{F}^*$), $m_{k,1} \in \mathcal{A}_{\pi_k,\pi_1}$ and H is a matrix in form of $\mathcal{M}_{k-1}$ from Lemma 6.2. This implies that elements present in H can be selected such that $\det(H) \neq 0$. Moreover, $\det(G) \neq 0$ as $\det(G)$ is the product of non-zero elements. If $m_{k,1}$ is not zero, then there exists a value $m_{1,1} = m_{k,1} \det(H)/\det(G) \in \mathbb{F}^*$ such that $\det(M) = 0$, and then the corresponding vectors of M would be linearly dependent, which contradicts the fact that $\mathcal{A}'$ (thus $\mathcal{A}$) is a SELIT solution. Therefore $m_{k,1}$ must always be 0, hence, $0 \in \mathcal{A}_{\pi_k,\pi_1}$, and from (6.9), this implies that $\mathcal{A}_{\pi_k,\pi_1} = \{0\}$.

From (6.13), implies that $\pi_k \not\prec \pi_1$ which concludes the proof by induction, hence the theorem. □

Theorem 6.3. Any canonical and restricted diagonal family $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ which is a solution of SELIT, is included in a family $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n)$ which is a permutation of lines and columns of the MILIC construction for n (that is $\forall i \in [\![n]\!], \mathcal{A}_i \subset \mathcal{B}_i$)

Proof. The binary relation $\prec$ is acyclic as a direct consequence of Theorem 6.2. We can use classical results to build a total order embedding it, see for instance [83]: let $\lhd$ be the transitive closure of $\prec$[3]. The transitive closure of an acyclic relation is irreflexive (see [83]).

Now, by the Szpilrajn extension theorem [84, pp. 386-389], for a transitive and irreflexive relation, there exists a total order, that includes it, denoted $\lll$. We can reorder the indices $[\![n]\!]$ as a sequence $\mathrm{L} = \{\pi_1, \pi_2, \ldots, \pi_n\}$ using the total order such that $\pi_1 \lll \pi_2 \ldots \pi_{n-1} \lll \pi_n$. The mapping of $\mathcal{A}$ with L (actually a permutation) is given by

$$\mathcal{P}(\mathcal{A}, \pi_1, \pi_2, \ldots, \pi_n) = \begin{bmatrix} \mathbb{F}^* & \mathcal{A}_{\pi_1,\pi_2} & \cdots & \mathcal{A}_{\pi_1,\pi_n} \\ \mathcal{A}_{\pi_2,\pi_1} & \mathbb{F}^* & \cdots & \mathcal{A}_{\pi_2,\pi_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}_{\pi_n,\pi_1} & \mathcal{A}_{\pi_n,\pi_2} & \cdots & \mathbb{F}^* \end{bmatrix}. \tag{6.15}$$

Consider any $\pi_i \in [\![n]\!], \pi_j \in [\![n]\!]$ and $\pi_i \neq \pi_j$. Assume that $\mathcal{A}_{\pi_i,\pi_j} \neq \{0\}$. From (6.13), $\mathcal{A}_{\pi_i,\pi_j} \neq \{0\}$ iff $\pi_i \prec \pi_j$. Then $\pi_i \prec \pi_j$ implies that $\pi_i \lhd \pi_j$ and consequently $\pi_i \lll \pi_j$. Then $i < j$ (because $\pi_i \lll \pi_j \iff i < j$), and this element $\mathcal{A}_{\pi_i,\pi_j}$ must be in the upper triangle of (6.15).

As a consequence, if π_i, π_j, is in the lower triangle (*e.g.*, $j < i$) then $\mathcal{A}_{\pi_i,\pi_j} = \{0\}$. This proves that the family $\mathcal{A}' = TP(\mathcal{A}, \mathrm{L})$ has a triangular matrix, hence is included in the MILIC construction for n, shown in (6.8). $\mathcal{A}'$ is obtained through a permutation of $\mathcal{A}$, hence the theorem, with $\mathcal{B}$ obtained through the inverse permutation of the MILIC construction for n. □

Now the most general form of the theorem is obtained by no longer considering restricted diagonal families, but any diagonal family:

Theorem 6.4. Any canonical and diagonal family $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ that is a solution of SELIT, is included in a family $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n)$ which is a permutation of lines and rows of the MILIC construction for n (where "included" means that $\forall i \in [\![n]\!], \mathcal{A}_i \subset \mathcal{B}_i$).

Proof. Let $(\mathcal{A}_{i,j})$ be the sets of coefficients in $\mathcal{C}(\mathcal{A})$. Let $\mathcal{A}'$ be the family defined from its sets of coefficients $(\mathcal{A}'_{i,j})$ from $\mathcal{C}(\mathcal{A}')$ selected as follows: $\forall i \in [\![n]\!], \forall j \in [\![n]\!]$ if $\mathcal{A}_{i,j} \neq \{0\}$ then $\mathcal{A}'_{i,j} = \mathcal{A}_{i,j} \setminus \{0\}$ otherwise $\mathcal{A}'_{i,j} = \{0\}$.

$\mathcal{A}'$ is now a restricted diagonal solution, hence Theorem 6.3 can be applied, and $\mathcal{B}$, a permutation of a MILIC construction can be found such that $\mathcal{A}'$ is included in $\mathcal{B}$. Now the only difference between $\mathcal{A}'$ and $\mathcal{A}$, is that vectors of transversals of $\mathcal{A}$ may have 0 in entries in positions where vectors of transversals of $\mathcal{A}'$ might not: but then they would still be included in $\mathcal{B}$, hence $\mathcal{A}$ is included in $\mathcal{B}$, hence the theorem. □

6.7 Alternative Algebraic SELIT Solutions

The proofs in the Section 6.6 give insights on the reasons why our constructions have specific structures. We also tried to explore some other families of solutions

[3] the transitive closure of a binary relation $\prec$ on a set is the smallest relation on the set that contains $\prec$ and is transitive

that are not canonical, component-wise and diagonal. An example of one such solution named *Matryoshka* is presented in this section. The name Matryoshka is given based on the way the sets are arranged in a table in a nested way. The sets are chosen such that they belong to subfields of the field $\mathbb{F}$.

Consider

$$M = \begin{bmatrix} x & a \\ b & c \end{bmatrix}.$$

We have

$$\det(M) = 0 \iff x = abc^{-1}.$$

When x can be selected freely from $\mathbb{F}^*$, we can make $\det(M) = 0$, unless one of a, b or c is zero. Notice alternately that if a, b, and/or c can take several values, the generalizations of the Cauchy-Davenport theorem [85, pp. 141-142] shows that abc^{-1} will take even more values, hence making the chances that $\det(M) = 0$ even higher, unless, for instance, they are in a multiplicative subgroup. This is the insight for proposing the following alternative example of SELIT solution.

Theorem 6.5. Let $H_0 = \{0\}$, and let $H_1 \subset \ldots \subset H_n$ be nested subfields of $\mathbb{F}$ and denote $\forall i \in [\![n]\!], C_i{=}H_i \setminus H_{i-1}$. Let $\mathcal{A}$ be the set family such that

$$\mathcal{C}(\mathcal{A}) = \begin{bmatrix} C_n & C_{n-1} & C_{n-1} & \cdots & C_{n-1} & C_{n-1} & C_{n-1} & C_{n-1} \\ C_{n-1} & C_{n-1} & C_{n-2} & \cdots & C_{n-2} & C_{n-2} & C_{n-2} & C_{n-2} \\ C_{n-1} & C_{n-2} & C_{n-2} & \cdots & C_{n-3} & C_{n-3} & C_{n-3} & C_{n-3} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ C_{n-1} & C_{n-2} & C_{n-3} & \cdots & C_4 & C_3 & C_3 & C_3 \\ C_{n-1} & C_{n-2} & C_{n-3} & \cdots & C_3 & C_3 & C_2 & C_2 \\ C_{n-1} & C_{n-2} & C_{n-3} & \cdots & C_3 & C_2 & C_2 & C_1 \\ C_{n-1} & C_{n-2} & C_{n-3} & \cdots & C_3 & C_2 & C_1 & C_1 \end{bmatrix} \tag{6.16}$$

Then $\mathcal{A}$ is a SELIT solution.

Proof. We prove this by induction. First pick an arbitrary transversal from $\mathcal{A}$, and write the matrix M_n of its vectors. Then one can write

$$\det(M_n) = a_n \det(M_{n-1}) + \ldots + (-1)^{n-1} a_1 \det(B_1)$$

where

$$a_n \in C_n, a_{n-1} \in C_{n-1}, \ldots, a_1 \in C_{n-1}$$

and M_{n-1} is a matrix with a similar form as M_n of size $(n-1) \times (n-1)$ and $B_1, \ldots, B_{n-1}$ are matrices with coefficients in H_{n-1}.

We can write the determinant of M_n as

$$\det(M_n) = a_n \det(M_{n-1}) + h$$

where $h \in H_{n-1}$. If

$$\det(M_{n-1}) \in H_{n-1},$$

and

$$\det(M_{n-1}) \neq 0,$$

then it is not possible to take

$$a_n = -\det(M_{n-1})^{-1}h \in H_{n-1}$$

because

$$a_{n+1} \in C_n = H_n \setminus H_{n-1},$$

hence $\det(M_n) \neq 0$, and still $\det(M_n) \in H_n$.

This is the basis for a proof by induction that $\det(M_n) \neq 0$, knowing that for $n = 1$ the property is true. Hence $\mathcal{A}$ is a SELIT solution. □

Note that given a finite field $\mathbb{F}$, the Matryoshka construction is only possible until a fixed n, which depends on subfields of $\mathbb{F}$. But for any n, one can select a finite field $\mathbb{F}$ that allows the construction. Recoding is possible with the Matryoshka construction. Vectors form sets defined by the bottom rows may be combined to obtain a vector form sets defined by the rows higher up in Table 6.16. Hence, the vectors from the sets defined by the bottom rows should be sent before the ones from the top.

To define Matryoshka construction for an n dimension SELIT solution one needs a field that includes n distinct subfields (including itself). One example is $\mathbb{F} = 2^{(2^n)}$ that includes $n + 1$ subfields. For example, consider $GF(256)$, that corresponds to $n = 3$, *i.e.*, $\mathbb{F} = 2^{(2^n)}$ the number of subfields possible is 4 (the subfields are $GF(256)$, $GF(16)$, $GF(4)$, $GF(2)$). Thus, the maximum generation size is 4. For larger generation size a larger n is required; however, the cardinality of the field grows very quickly with n. Consequently, the number of bits required to represent each element of the field can become impractical. For example, one can select $n = 10$ for a very limited generation size 11. Each element for the field requires 128 bytes. 1280 bytes are required for a vector of generation size 10, which is exactly the minimum MTU of IPv6.

Example 6.7. Alternatively, we can define a family of sets that is a combination of MILIC and Matryoshka, for instance as follows

$$\mathcal{C}(\mathcal{A}) = \begin{bmatrix} C_4 & C_3 & C_3 & C_3 & \mathbb{F} & \mathbb{F} & \mathbb{F} & \mathbb{F} \\ C_3 & C_3 & C_2 & C_2 & \mathbb{F} & \mathbb{F} & \mathbb{F} & \mathbb{F} \\ C_3 & C_2 & C_2 & C_1 & \mathbb{F} & \mathbb{F} & \mathbb{F} & \mathbb{F} \\ C_3 & C_2 & C_1 & C_1 & \mathbb{F} & \mathbb{F} & \mathbb{F} & \mathbb{F} \\ 0 & 0 & 0 & 0 & C_4 & C_3 & C_3 & C_3 \\ 0 & 0 & 0 & 0 & C_3 & C_3 & C_2 & C_2 \\ 0 & 0 & 0 & 0 & C_3 & C_2 & C_2 & C_1 \\ 0 & 0 & 0 & 0 & C_3 & C_2 & C_1 & C_1 \end{bmatrix}. \tag{6.17}$$

Any matrix M constituted of the vectors from (6.17) is a block matrix of the form

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

Since C is a null matrix, M is a triangular block matrix

$$M = \begin{bmatrix} A & B \\ 0 & D \end{bmatrix}.$$

The determinant of a triangular block matrix is equal to the product of the determinant of its diagonal blocks [86]. A and D are Matryoshka matrices and $\det(A) \neq 0$ and $\det(D) \neq 0$ from Theorem 6.5. The family of sets in (6.17) is a SELIT solution of dimension 8 that requires only 4 subfields. This can be generalized for any Matryoshka construction and any MILIC construction. However, the re-coding is guaranteed only among the elements of upper or lower blocks. Indeed, in general, linear combinations of vectors from the lower block would not give vectors from the upper block and vice versa. Thus, from a re-coding perspective, it is restrictive and can lead to inefficiency especially at the end of content retrieval.

We also tried exploring other SELIT solutions by searching through the sets of all the possible vectors for a specific field using computer. One example solution is presented in Example 6.8 for field $\mathbb{F} = GF(5)$ and $n = 3$. The resulting sets of vectors are neither canonical, component-wise nor of the matryoshka form.

W

Example 6.8. Consider the Field $\mathbb{F} = GF(5)$ and $n = 3$. One example of sets of vectors that are a SELIT solution found by exhaustive search is given below.

$$\begin{aligned}
\mathcal{A}_1 = \{&(0,1,1)\ (1,0,3)\ (1,3,2)\ (1,0,0)\ (1,4,4)\ (0,1,2)\ (1,3,1)\ (0,1,3)\ (1,0,1)\\
&(1,3,0)\ (1,4,2)\ (0,1,4)\ (1,1,1)\ (1,4,3)\ (1,0,2)\ (1,1,0)\ (1,0,4)\ (1,4,0)\\
&(1,1,3)\ (1,4,1)\ (1,1,2)\ (1,3,4)\ (0,1,0)\ (1,1,4)\ (1,3,3)\}\\
\mathcal{A}_2 = \{&(1,2,2)\ (0,0,1)\ (1,2,4)\}\\
\mathcal{A}_3 = \{&(1,2,1)\ (1,2,0)\ (1,2,3)\}
\end{aligned}$$

The sets $\mathcal{A}_1, \mathcal{A}_2$, and $\mathcal{A}_3$ form a solution to a dimension 3 SELIT problem. We can find some solutions by exhaustive search but they are computationally expensive to find and cannot be generalized.

6.8 Conclusion

In this chapter we formally introduced SELIT as a problem specific to the network coding-based NDN scenario. We investigated a special group of families of sets (canonical, component-wise and diagonal) that are solutions to this problem. We were able to prove that a large class of such families of solutions for $|\mathbb{F}| \geq 3$, where one can pick the coefficient vectors from fixed sets and with additional constraints, are essentially a version of the MILIC construction presented in Chapter 4. We presented an alternate Matryoshka construction and an example of SELIT solution found by exhaustive search.

We can conclude that it appears to be difficult to find very general families that are not isomorphic to MILIC. The Matryoshka construction is interesting, but one main issue is that the number of bits required to represent the elements of the field grows quickly with the generation size n[4]. If this issue is not addressed it limits the dimension of generation size for the content unlike MILIC

[4] exponentially if $\mathbb{F} = 2^{(2^n)}$

to $n < 10$ in practice. Hence, MILIC appears to be more practical solution for the SELIT problem in context of network coding-based NDN.

The results presented in this chapter are mostly specific to canonical families and leave an open question about the non-canonical families (that do not include the canonical vectors). Also there is an open question if other solutions for such problem can be proposed (*e.g.,* using coding theory results). Another perspective for the future work is that the problem can also be generalized as k-SELIT for finding $k < n$ sets instead of n.

7

Conclusion and Future Perspectives

The growth of Internet applications, the increased number of connected devices, and the users' ability to easily create, distribute, and consume content over the Internet have motivated the development of new methods to access data. Information-Centric Networking is one of the network architectures proposed to cater to skyrocketing content production and consumption. NDN focuses on the content rather than its location. NDN enables content caching anywhere in the network and does not depend on location-specific connections, naturally supporting multi-path data delivery.

Network coding has been shown to bring many benefits to ICN in terms of fast data delivery [6, 53]. Network coding allows the nodes to benefit from multi-path retrieval by allowing forwarding and processing multiple Interests in parallel. It also helps increasing content diversity by storing coded content enabling the nodes to serve more requests. Nevertheless, supplementing NDN with network coding raises several issues, which have been partly addressed in this thesis. The following sections present the main contributions of the thesis. Later we propose some future directions to further improve the integration of network coding in NDN architectures.

7.1 Main Contributions

When supplementing NDN with network coding, Interests request coded content. It is necessary to ensure that each Interest sent by a client in the network brings new information to efficiently retrieve coded content in NDN. The solutions presented in the past that ensured linear independence between Data

packets had shortcomings of considerable overhead in Interest packets. The first contribution of this thesis, presented in Chapter 3 is to provide an effective way to reduce this overhead. We proposed a coding approach to reduce the size of the Interest packets when requesting network-coded packets. This method effectively reduces the overhead and allows network nodes to quickly check if the content available in their cache would be linearly independent to the requesting node's content. The thesis then presents a detailed analysis of NetCodCCN, a family of previously proposed protocols integrating ICN and network coding that tend to reach the network capacity.

The main contribution of this thesis presented in Chapter 4 is MICN, a network coding-based NDN protocol. MICN permits the client nodes to implicitly state what they require to download a generation by a specific naming scheme called MILIC. MILIC partitions the set of all possible encoding vectors into subsets. The subsets are defined such that if one vector is drawn randomly from each of these subsets, the resulting set of vectors are always linearly independent. MILIC identifies these subsets with indices. The client nodes add the index in Interest packets that imposes constraints on the coded segments that respond to these Interests. A coded segment that can serve as response to the indexed Interest has encoding vectors chosen randomly from pre-defined MILIC subsets. The syntax of MICN Interests allows them to be sent and processed in parallel, effectively allowing the network to achieve its maximum capacity.

The evaluations showed that MICN brings significant improvements in terms of download delay compared to the classical NDN. The fast content retrieval of MICN, however, comes at the cost of some redundant data traffic. The thesis also proposed two solutions to reduce this redundant traffic. The first consists of supplementing MICN with an Interest Cancellation scheme in a protocol variant, MICN-IC. MICN-IC cancels the pending Interests in the network whose corresponding content is no longer required at the client. The cancellation is achieved by adding a single vector in the Interest packet as feedback to the network. The additional vector has a dimension equal to the generation size. MICN uses a multicast forwarding strategy that results in some flooding of the Interests and incurs this redundant traffic. The second solution for this problem proposed in this thesis consists in the proposal of several modified forwarding strategies. Chapter 5 details and analyze these modified forwarding schemes. These modified forwarding techniques dynamically change the forwarding strategy of a node during content retrieval to reduce the number of Interests in the network and reduce redundant data traffic.

MILIC requires the partition of network coding vectors (and of associated coded Data packets) into subsets such that they give linearly independent transversals, where *transversals* are obtained by drawing one random vector from each subset. Motivated by the MILIC property of getting linearly independent transversals, the last contribution is a detailed analysis of families that can give partitions of sets with linearly independent transversals. The main result of this contribution is that direct generalizations of the MILIC family turn out to be equivalent (isomorphic) to MILIC. Some alternate constructions are also proposed, but, unfortunately they appear to have limitations in terms of practicability in MICN context.

The architecture presented in this thesis is implemented in our own Python

simulator that implements core NDN semantics. The simulator uses the guidelines of NDN provided in the NFD developer's guide [34]. The architecture is easily adaptable over other Interest-based ICN architectures.

7.2 Perspectives

This section presents some future research directions and perspectives related to some key aspects of network-coded ICN architectures that were not addressed in this thesis and some aspects that require further investigations and improvements.

7.2.1 Sliding Window Coding

The content in MICN is divided into generations to reduce the download time and decoding complexity. The dimension of generations is decided at the source, and the network coding operations performed throughout the network respect the generations. Nevertheless, as observed in Section 5.2, content from one generation can affect the download time of the next generation. It would be interesting to explore sliding window [54] coding schemes. Such coding schemes would not have solid boundaries between generations and make it possible to group all content segments in a single generation. When using sliding window coding, the coding window size w decides on the number of source segments that can be coded together. In MICN, original segments of a generation are coded based on MILIC constraint defined in (4.1). In the sliding window MICN context, starting from a i-th segment at most w consecutive original segments can be coded together. We experimented with the sliding window approach applied to MICN without any optimization and obtained similar download time results to MICN with multiple generations (a gain of about 2%). This still does not close the gap compared to MICN with optimizations (MICN-IC). Thus, one still needs to supplement the sliding window approach with additional mechanisms to improve performance. For instance, on one tested scenario on the butterfly topology, MICN-IC with multiple generations is within 11% of the max-flow, MICN with sliding window is within 60%, and finally, MICN-IC with sliding window is within 0.1%. Sliding window coding appears promising, and further work is required to find the best way to integrate it with MICN.

7.2.2 Caching

In this thesis, the caching capacity of the nodes was always considered to be unlimited or large enough to cache all packets of a content generation. We did not explore the impact of caching strategies and cache eviction on the performance of MICN. Nevertheless, this is not the case in real-life networks. The caches have a limited capacity; not every content is cached, and the cached content is evicted after some time. The decision to cache content based on its popularity at the right place in the network (*e.g.,* at the edge of the network) can reduce the delivery delay and consequently increase the throughput in ICN [39, 3] and network-coded ICN [87]. It would be interesting to develop efficient cache management policies in coordination with the MICN protocol. The goal of such

caching policy is to take into account the limited capacity of caches while still making it feasible to perform network coding and increase the cache hit rate.

7.2.3 FIB Management

The information from the Forwarding Information Base allows the Interests to be forwarded to the potential source(s) of data. In MICN, the multicast forwarding along with network coding helps improve content dissemination in a network without fully developed or functional FIB. However, it may lead to inefficient use of the network resources, *e.g.,* the lingering Interests and consequent redundant data traffic in MICN. During this thesis, the positions of the source and client were considered fixed. Nevertheless, a client can be mobile in a real network, which is an additional challenge to address.

Additionally, due to the ICN distributed caching infrastructure, the availability and location of the content may also vary over time. It is a consequence of temporary copies of the content cached in the network that may be removed based on caching policies. The MILIC Interests ensures that the clients receive linearly independent content, no matter where it is cached in the network. However, an improved FIB management protocol that updates the FIB in case of mobility or cache eviction can help identify slow and unresponsive faces. It would be interesting to have a self-learning FIB management that allows the network nodes to update information online during content retrieval.

7.2.4 Security

Security issues, including privacy, authenticity are not addressed in this thesis. Most of the network coding approaches presented in the past do not consider this aspect as well.

ICN does not consider the localization of the content; however, the authenticity of the content is essential. The retrieved content has to be from a trusted source. Security is an important research topic in ICN [32], and some approaches have been proposed for content security in ICN [88]. In ICN, each content object is encrypted and signed by the source, and the client can verify the authenticity of the content upon retrieval.

Network coding inherently provides content security since an eavesdropper cannot reproduce the original content until it has enough linearly independent packets. Nevertheless, packets are mixed with network coding in ICN, *i.e.,* a new packet is generated with a new signature independent of the producer. The signature invalidates the producer's signature making it difficult to authenticate the content at the client and making it vulnerable to security attacks, *e.g.,* pollution attacks [89]. A suitable distributed trust model [90] is required to sign network-coded content packets at the intermediate nodes.

Secure Practical Network Coding (SPOC) [91] is an approach that encrypts a subset of encoding coefficients of each Data packet, with keys available at the source and the client, but not at the intermediate nodes. A subset of unencrypted coefficients allows network coding in the network. However, decoding is only possible after the encrypted coefficients are decoded.

A homomorphic signature scheme has been proposed for network coding

authentication [92, 93]. A network-coded NDN authentication model based on the homomorphic signature scheme have been explored in [94, 95, 96]. However, they do not provide implementations for their schemes in actual network-coded NDN scenarios.

Security in network coding enabled ICN architectures is an open research question. It would be interesting to implement one of the aforementioned network coding security mechanisms in the context of MICN.

7.2.5 Congestion Control

In this thesis, we focus on reaching maximum throughput, assuming one type of connection and not addressing congestion. However, multiple paths with multiple types of connections in a real network and multiple clients requesting content can saturate the network. It would be necessary to regulate the streams of Interests and Data packets, which requires some congestion control. There is a large amount of work in the past addressing congestion control and ICN; for instance, see [97, 98, 99, 96] and some on congestion control in the context of network coding [100]. It would be interesting to have a fully deployed solution on an actual network to study congestion in network-coded ICN architectures.

7.2.6 SELIT Solutions

The solution of the SELIT problem has been studied on a class of solutions that are very similar to MILIC construction, *i.e.,* canonical component-wise and diagonal. The thesis also presents some alternate constructions. However, only preliminary analysis on the practicality of the alternate constructions has been performed. Because of the link between the SELIT problem and efficient network coding over ICN, it would be interesting to first conduct a more thorough analysis, and second to find and explore more diverse families of solutions to the SELIT problem. It might be possible to construct SELIT solutions from superimposed codes [101] or other coding constructions.

7.2.7 Adaptive Forwarding Strategies

For content retrieval, MICN is efficient even in the presence of losses, as illustrated through the example scenarios in Chapter 4. This performance gain, however, can be interpreted as a consequence of the Interest flooding. Chapter 5 presents some forwarding techniques integrated with MICN implemented on nodes to address the Interest flooding and redundant content overflow problem. All the techniques are presented after studying the network behavior and the information readily available at the nodes, *e.g.,* the pending queues on different faces and the content available in their Content Stores. These techniques exploit this information to decide if a node should forward a received Interest and if the node should duplicate it on some or all of the available faces. These heuristic-based modified forwarding techniques attempt to achieve the following goals:

1. Ensuring the retrieval of a decodable set of coded content.

2. Ensuring that the retrieval of the decodable set is achieved at maximum speed (maximum throughput).

3. Reducing the number of lingering Interests in the network.

4. Reducing the number of redundant data traffic.

Each of the above goals may result in contradicting forwarding decisions. For example, if a node aims to reduce the number of lingering Interests in the network, it can reduce the number of Interests it duplicates on its available faces. The redundant content will consequently be reduced due to the limited number of Interests forwarded in the network. However, the multiple available paths will no longer be exploited resulting in an increased download time. Additionally, if the nodes do not forward enough Interests, this may result in the client not getting back a decodable set of coded content, *e.g.,* in case of losses, *etc.*

The modified strategies helped curb the problem of flooding and redundant traffic. Nevertheless, the price to be paid is reduced robustness to packet losses. Some forwarding strategies may not be able to adapt in case of losses or other network topology changes. In real network scenarios, varying bandwidth, connectivity, network losses, and even user mobility may change the network dynamics. More dynamic forwarding techniques are required for a node to make forwarding decisions that ensure maximum throughput and avoid flooding [4].

When looking for dynamic and adaptive forwarding solutions, it is natural to explore other methods such as machine learning. Indeed, reinforcement learning techniques have been proposed in the past to improve the Interest forwarding in ICN, by using Multi-Armed Bandit strategies [102, 103], Q-learning approaches [104, 105] and other approaches. This family of works mostly focuses on finding the one best path for Interest forwarding in classical ICN architectures.

Thus, a promising direction is to adopt reinforcement learning techniques to automatically train adaptive forwarding strategies in MICN, instead of using hard-coded heuristics. With MICN, we observe that the problem appears to require powerful techniques mostly due to the credit assignment problem [106]; because it is hard to identify which forwarding decision is responsible if some of the previously listed goals is not met.

We conducted some work during the course of this thesis to experiment with various reinforcement learning techniques in MICN. The outcome of this work is presented as a summary of the models, training methods, preliminary results, and related details in the Appendix A.

A

Reinforcement Learning for MICN

As described in Section 7.2.7, the heuristic-based forwarding strategies improve network resources utilization in MICN face some shortcomings. For example, the parameters are hardcoded and require tuning for changes in the network state, *e.g.,* in the presence of losses. Our aim in this appendix is to explore adaptive forwarding solutions based on reinforcement learning algorithms.

A.1 Reinforcement Learning

Learning is a process to improve a system's performance based on experience. The improvement comes from information about the actions taken in certain situations and their outcomes in the past. This information is then utilized to adjust the future decision-making of a system.

Reinforcement Learning (RL) allows mapping situations to actions and aims at maximizing a numerical reward. RL involves an *agent* that interacts with the environment. It receives stimuli from the environment defining its *state* and then reacts by choosing an *action*. An agent tries different actions to learn the best actions for each state. The actions taken in any state are then evaluated as a *reward* towards a set goal. The rewards may be positive or negative. The agent uses the observed rewards to improve its actions in the future, *i.e.,* take actions that bring better rewards and help the agent reach the set goals. Additionally, the agent learns to avoid actions that are unfavorable for it to reach its final goal. The actions taken in different states by the agent are called a *policy*.

In general, policies may be deterministic*, i.e.,* specifying a unique action per state, or stochastic*, i.e.,* specifying probabilities for each action [107]. RL is often used for networking; it helps with algorithms, decisions, and policies, *etc.*

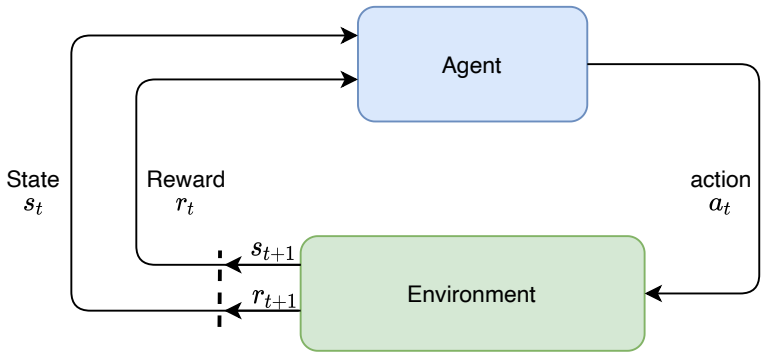


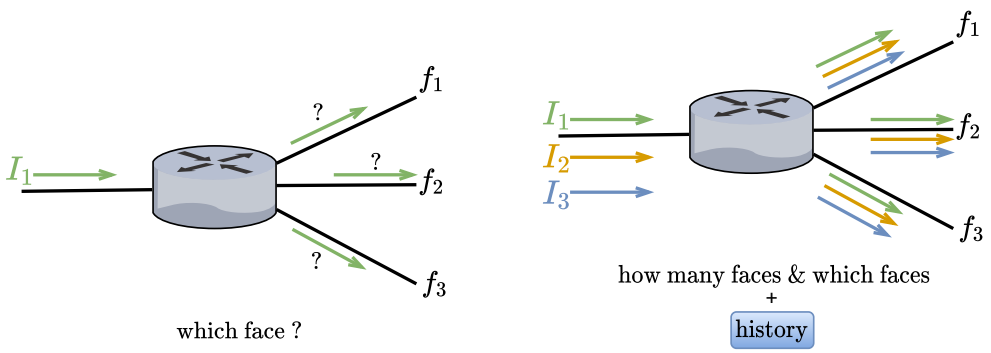
Figure A.1 – Reinforcement Learning model

RL algorithms can learn and adapt the forwarding in MICN over generations to improve the performance.

A.2 Reinforcement Learning based Forwarding in ICN

The ICN literature proposes some works implementing RL techniques to improve forwarding in different ICN networks. Chiocchetti *et al.* [104] propose INFORM, a dynamically changing forwarding mechanism for ICN. Inspired by the Q-routing framework [108], INFORM uses RL to discover paths to temporary copies of content not addressed in routing tables. Avrachenkov *et al.* [102] present the ICN Interest forwarding problem as a Multi-Armed Bandit (MAB) problem with delays. Three well-known algorithms are implemented in [102], namely ε-greedy, tuned ε-greedy, and UCB (Upper Confidence Bound) to minimize the number of Interests forwarded to suboptimal routes. Bastos *et al.* [103] also proposed a Multi-Armed Bandits Strategy (MABS) algorithm: an ε-greedy algorithm is used to explore the network for better paths. The content routers build their FIB by keeping a record of each interface retrieval time related to a specific content name. Abane *et al.* [109] propose a forwarding strategy for low-end IoT. A cost is associated with each packet; the nodes overhear packets and learn a cost value by reinforcement learning. The nodes decide to forward an Interest with a delay according to their cost-based eligibility. Fu *et al.* [105] propose an adaptive forwarding scheme in ICN using a Q-learning-based protocol. They propose IQ-Learning (Interest Q-Learning) and DQ-Learning (Data Q-Learning) strategies that learn from the past choices to make the best delay-efficient forwarding choices. Other literature using RL techniques to improve Interest forwarding in ICN may be found in [110, 111]. A survey on routing approaches for networks using RL is presented in [112]. A Monte-Carlo Tree Search (MCTS) algorithm is proposed in [113] to construct the path trees from the users to the sources.

In traditional NDN forwarding, a node decides the path in terms of the next hop where an Interest should be forwarded. Most of the works found in the literature are working towards this goal, *i.e.,* to find the correct path to forward their Interests. These works evaluate the cost in terms of content availability and delay to learn the most feasible or delay-efficient path to the content. If



(a) ICN (b) Network-coded ICN

Figure A.2 – Interest forwarding decision

more than one path is available, these algorithms enable nodes to choose the best paths to forward their Interests. However, it is essential to recall that each Interest in classical ICN corresponds to one content. Each learning algorithm should ideally choose one path for each Interest, as presented in Figure A.2a. The content from one Interest does not affect the next or previous Interests and, consequently, future forwarding decisions.

With network coding, since one Interest can be answered by a large variety of linearly independent content, Interests may be duplicated on several paths in order to take advantage of the different coded content cached in the network. Consequently, the previously sent Interests may impact the Interest forwarding decisions of the following Interests based on the content received. As a result, the Interest forwarding decisions with network coding are different and more complex than Interest forwarding for uncoded content. As shown in Figure A.2b, the decision in network coding scenario includes the number of faces and the exact faces for forwarding Interest. Additionally, each forwarding decision also needs to take into account the previous forwarding decisions. An example scenario is presented in the expected Rank-based strategy in Section 5.3.5. Each forwarding decision is taken based on the previous decisions and the content received in response to those decisions.

The classical NDN forwarding strategies are either flooding-based, in which case Interests are broadcast everywhere, or route-dependent, in which case essentially one path is chosen. Both strategies have some unwanted consequences in the network coding context. The flooding strategy incurs extra data traffic, while route-dependent strategies limit the ability to exploit all available paths, degrading the performance to the equivalent of uncoded NDN.

A.3 Adaptive Forwarding in MICN using Reinforcement Learning

MICN implements a multicast strategy to exploit all available paths, as its main focus is to achieve high throughput. Since multicast is a flooding-based strategy, it results in redundant traffic. Classical ICN forwarding strategies are insufficient for MICN. The forwarding strategies should decide on a) how many

faces and b) which faces to forward Interests to (as presented in Figure A.2b); this may require taking into account the history of forwarding decisions for a particular content generation. Ideally, dynamically adaptive forwarding strategies are required to utilize the network resources more efficiently, and to adapt their variations.

As observed with modified strategies in Chapter 5, a mix of a non-conservative strategy, *e.g.,* multicast, and a conservative strategy that limits Interest forwarding, is required to ensure high throughput and avoid redundant data traffic. We will try to use RL to find this trade off.

Based on the four goals presented in Section 7.2.7, the main performance metrics for adaptive forwarding techniques in MICN are:

1. Retrieve a decodable set of content generation.
2. Getting back content at the maximum possible rate.
3. Avoid redundant data traffic.
4. Limit the number of Interests sent in the network.

In the threshold-based forwarding strategy (see Section 5.3.1), we observed that the nodes multicast Interests only at the beginning of content retrieval. Albeit the threshold-based forwarding failed to achieve maximum throughput, it reduced the redundant traffic considerably. The redundant data traffic is reduced because the nodes stop forwarding Interests later in content retrieval. Other modified techniques limit the Interest forwarding; however, based on their tuning parameters, they achieved one goal but had to compromise another partly. Based on this observation, in order to achieve the performance goals, MICN works well with a forwarding strategy that:

- exploits multi-path content retrieval of NDN (forwards Interests on multiple faces to encourage fast content retrieval) and
- switches to conservative forwarding (towards the end of a generation retrieval or if too much content is already expected).

In order to build a forwarding policy considering the progress in content retrieval as a state variable may be useful, instead of continuing with a static policy or having static parameters for modifying the strategy. Our goal with RL techniques is to allow the nodes to update their forwarding strategy dynamically, unlike the modified techniques presented in Chapter 5.

The idea is to implement an RL algorithm independently on each node in the network. The client nodes try to retrieve a content generation; the retrieval of one content generation defines one episode. Each network node learns from the outcomes of the policy implemented in one episode to improve the policy in the next, based on how fast it can retrieve content and how much redundant content is received at the node. The nodes also evaluate if any useless Interests were forwarded, *i.e.,* no content was received for a particular Interest because there is no content source or cache on the forward link.

For our particular problem, the state can be the progress in a generation of the requested content, *i.e., the* rank of the CS or the number of received

innovative content. However, in general, the state can also be the whole history and observations of the Interests received, forwarded and pending, the content received, *etc.*

The action space $\mathcal{A}$ includes the number of faces and the set of possible faces to forward an Interest to, including the set of all available faces and not forwarding an Interest at all. Following the goals, actions that bring back innovative content in the least amount of time are rewarded, and those that bring back redundant or no content are penalized (negative reward). Moreover, a significant penalty is associated at the end of all actions if the client node fails to retrieve the entire generation.

We initially tested a Q-learning method to learn the best forwarding action. We implemented the Monte-Carlo algorithm [107, p. 101] over episodes of content retrieval. With Q-learning, we observed sudden changes in policies. Q-learning works with non-stochastic policies, and the action with the best action-value is chosen (almost) always. This leads to oscillations in the learning process even with epsilon-greedy methods. Additionally, in a complex network with multiple nodes or agents, one agent's decision affects the actions of other agents. The multi-agent scenario and the impact of past forwarding decisions on future ones give rise to a credit assignment problem: it is hard to assess which action is responsible for the goal not being met. Based on these difficulties, instead, we wish to have a stochastic policy that assigns probabilities to each action in each state.

A.3.1 Policy Gradient Methods

Policy gradient methods [107] are a type of RL techniques that rely upon optimizing *parametrized policies* with respect to the expected return (long-term cumulative reward) by gradient ascent. Action-value methods learn the values of actions and then select actions based on their estimated action values. However, policy gradient methods are not action-value methods, and they learn a parameterized policy that can choose actions without a value function. The value function is used to learn the policy parameters but is not directly involved in action selection.

REINFORCE [107, p. 330] is a policy gradient method that works for episodic tasks, *i.e.,* the algorithm generates a complete episode and uses the returns to compute the gradient. The state, action, and reward at each time $t \in 0, 1, 2, ...$ are denoted $S_t \in \mathcal{S}$, $A_t \in \mathcal{A}$, and $R_t \in \mathcal{R}$ respectively. Recorded episode is a sequence or trajectory of states, actions and rewards like

$$S_0, A_0, R_1, S_1, A_1, R_2, \dots .$$

REINFORCE is a plain policy gradient method and is suitable for stochastic policies. It computes a stochastic approximate gradient. It is considered as a policy gradient variant of the Monte-Carlo methods [107, p.101]. Monte-Carlo methods apply to episodic tasks and estimate the value function for each state and action pair. They do so by updating these estimates after completing an episode and computing the rewards for each state-action pair.

The REINFORCE uses the Monte Carlo method returns from time t until the end of an episode, but instead of updating the value function, it updates

the policy. $\pi(a|s,\boldsymbol{\theta})$ refers to the policy which is parameterized by the policy's parameter vector $\boldsymbol{\theta}$. The policy can be represented as

$$\pi(a|s,\boldsymbol{\theta}) = \Pr\{A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$$

for the probability that action a is taken at any time t if the system is in state s at time t with policy parameter $\boldsymbol{\theta}$. The policy $\pi(a|s,\boldsymbol{\theta})$ should be differentiable with respect to its parameters, *i.e.,* $\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})$ has to exist for all θ.

In this part, we chose a policy gradient method that is a contextual gradient bandit. $\mathcal{S} = \{1,\ldots,k\}$ is the discrete state space. For each state $s \in \mathcal{S}$ a parameterized policy has a vector of parameters $\boldsymbol{\theta}_s$ for each action in the discrete action space $\mathcal{A} = \{1,\ldots,n\}$

$$\boldsymbol{\theta}_s = \begin{pmatrix} \theta_{s,1} \\ \vdots \\ \theta_{s,n} \end{pmatrix}.$$

All actions are assigned probabilities to be chosen in any state based on the preferences, according to an exponential soft-max distribution:

$$\pi(a|s,\boldsymbol{\theta}_s) = \frac{e^{\theta_{s,a}}}{\sum_{b=1}^{n} e^{\theta_{s,b}}}. \tag{A.1}$$

REINFORCE algorithm initializes the policy parameter $\boldsymbol{\theta}_s$ at random and generates an episode on policy

$$\pi(\boldsymbol{\theta}_s) : S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T.$$

Then, for $t = 0, 1, \ldots, T-1$ the return G_t is estimated as

$$G_t = \sum_{k=t}^{T} \gamma R_k \tag{A.2}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate that determines the present value of future rewards. If $\gamma = 0$, then only immediate rewards are considered, however as γ approaches 1 the return takes future rewards into account more strongly. REINFORCE estimates the returns G_t from actual sample trajectories and uses it to update the policy parameters at each time t as

$$\boldsymbol{\theta}_s^{(t+1)} = \boldsymbol{\theta}_s^{(t)} + \alpha G_t \frac{\nabla_{\boldsymbol{\theta}_s}\pi\left(A_t|S_t,\boldsymbol{\theta}_s^{(t)}\right)}{\pi\left(A_t|S_t,\boldsymbol{\theta}_s^{(t)}\right)} \tag{A.3}$$

where α is the learning rate. For our choice of $\pi(a|s,\boldsymbol{\theta})$

$$\frac{\nabla_{\theta_s}\pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})} = \begin{pmatrix} -\pi(1|s,\boldsymbol{\theta}) \\ \vdots \\ 1-\pi(a|s,\boldsymbol{\theta}) \\ \vdots \\ -\pi(n|s,\boldsymbol{\theta}) \end{pmatrix}.$$

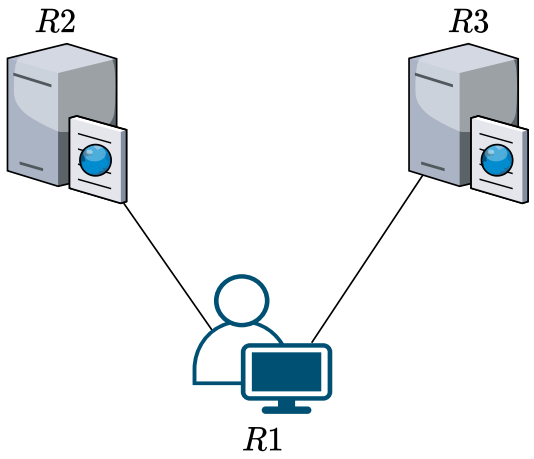


Figure A.3 – Topology used for Learning Algorithms

We implemented the REINFORCE algorithm from [107, p. 330] with and without the baseline. A baseline can be a function or a random variable that can be subtracted from the returns. Baseline subtraction does not affect the expected value of returns but reduces the variance and is expected to allow faster learning.

A method can use a learned value function $\hat{v}(S_t, \mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^d$ is a weight vector learned by the value function. REINFORCE uses a Monte Carlo method to learn the state-value weights $\mathbf{w}$ at the same time as $\boldsymbol{\theta}$. We choose the simplest value function to use in the baseline, for each state $s \in S$ a single scalar w_s is associated to each state s, *i.e.,* $\hat{v}(S_t, \mathbf{w}) = \mathrm{w}_s$. The value function parameter is updated after each policy's returns are estimated as:

$$\mathrm{w}_s = \mathrm{w}_s + \alpha\delta\mathrm{w}_s$$

where $\delta = G_t - \mathrm{w}_s$ and α is the same as the learning rate for $\boldsymbol{\theta}_s^{(t)}$.

A.3.2 Results for Policy Gradient Methods

We implemented a basic topology to test the policy gradient method, including a client node $R1$ and two source nodes $R2$ and $R3$ (See Figure A.3). The REINFORCE algorithm is implemented on the client node with two faces connecting to sources for the given topology. The node tries to retrieve a decodable set of content with generation size 20 at max flow and avoids redundant content. Each source stores a complete set of 20 segments. The pipeline size considered for this scenario is 2.

The state is identified as the progress within the generation, *i.e.,* the rank or the number of received innovative content. Given the generation size, there are 21 possible state values ranging from 0 to 20. The state space is quantized to two states. An initial state goes from no content to half the content received. The second state from half the content until the entire generation is received. The client application stops forwarding Interests after all the content is received.

For the considered topology, the actions vary from forwarding on one of the available faces ($R2$ or $R3$) to forwarding on all faces ($R2$ and $R3$) to forwarding to none. No link losses are considered. We do not implement the

timeout of Interests, *i.e.,* the clients do not resend Interests if content for an Interest is not received after the interest timeout.

Since REINFORCE works on episodic tasks, a reward is assigned after the end of an episode. A reward is computed based on the goals presented in Section 7.2.7. The reward R considers the rank, *i.e.,* the innovative content C_{innov}received compared to the expected final rank, *i.e.,* the generation size $\boldsymbol{g}$. Additionally, the reward takes into account the redundant content C_{red}, the innovative content retrieval time T_{innov}, and the amount of useless Interest forwarded I_{useless}, *i.e.,* the number of Interests forwarded that did not bring any content. Collectively, the reward R is computed as a weighted sum of the different objectives, and we found that the following expression gave good results

$$R = \left((C_{\text{innov}} - \boldsymbol{g}) - \eta C_{\text{red}} - 2\eta T_{\text{innov}} - \frac{1}{16}\eta I_{\text{useless}} \right) \times \frac{1}{\boldsymbol{g}}.$$

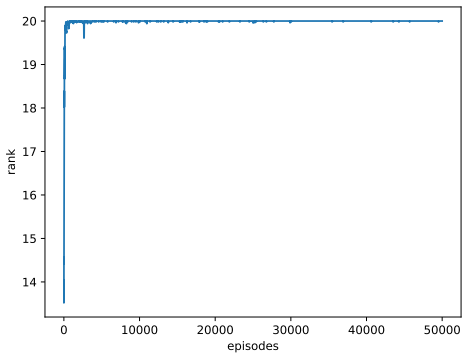
Additionally, a large penalty is inflicted if the client fails to retrieve a decodable set of content. The learning is performed over 50000 episodes. The parameter η is used to tune the reward. We consider a non-discounted case for return computation, *i.e.,* $\gamma = 1$ in (A.2). The performance is measured based on retrieving one generation, the time required to download, and the amount of redundant content received by the node in the network.

Figure A.4 presents the results of the REINFORCE algorithm. Figure A.4a shows the rank received by the client node $R1$ after each episode. In Figure A.4a, each point presents a moving average of 30 samples[1]. Rank indicates the innovative content packets of the generation received at the client. When 20 innovative packets are received, the client can decode the generation. Since the clients do not resend Interests in case of a timeout, a bad forwarding decision can result in the client failing to retrieve a decodable set. In such a case, a significant penalty is given to all the actions.

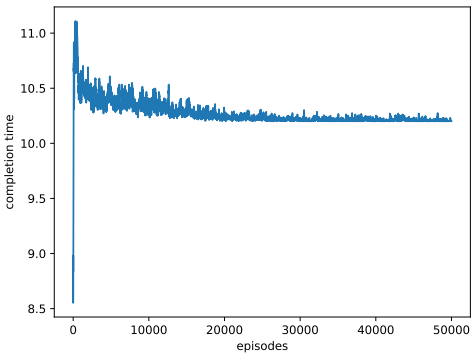
Figure A.4b indicates the time at which a decodable generation is retrieved at the client node. Figure A.4c represents the redundant content in the network over episodes. Figure A.4d plots the average return of each episode. These figures indicate that the learning algorithm is working well in terms of generation retrieval, completion time and reduction of the redundant content in the network.

Figures A.4e and A.4f present the policy of node $R1$ in the state 0 and 1, respectively, and its evolution over the successive episodes. The stochastic policy assigns a probability to each action for each state. The action itself is the set of routers where the Interests are forwarded ("$-$" means no forwarding). The policy in the first state tends to multicast the Interests by forwarding them on both available faces. However, the policy in the second state prefers action zero, *i.e.,* not sending to any faces. Since both faces of node $R1$ are connected to the sources, the duplicated Interests are both likely to bring two linearly independent content. The policy then limits the Interest forwarding to one or no face. The policy in both states collectively ensures that a decodable set of coded content is received with maximum throughput. In this case, the min-cut

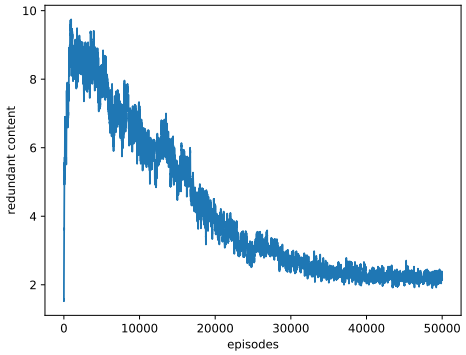
[1]Figures A.4b, A.4c, A.4d, A.5a, A.5b, A.5c, A.5d, A.5d, A.7a, A.7b, A.7c and A.7d all plot a moving average of 30 samples.



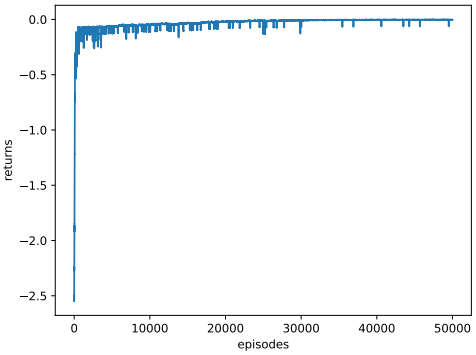
(a) Rank C_{innov} at the client node $R1$



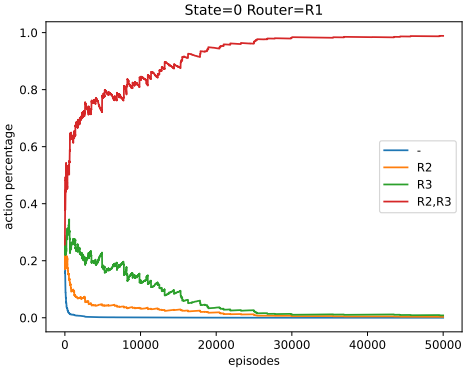
(b) Completion time T_{innov} of content retrieval at the client node



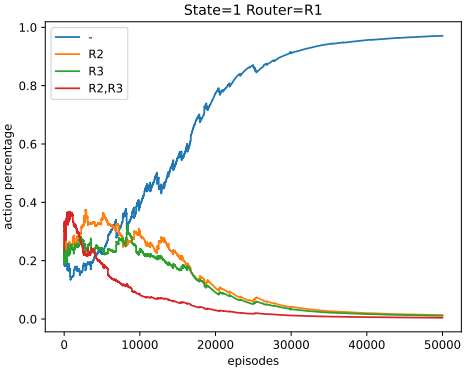
(c) Redundant content C_{red}



(d) Returns G

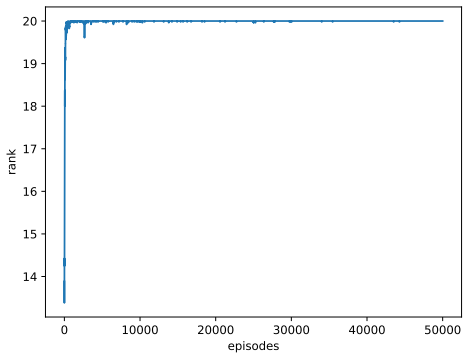


(e) Policy at the client note $R1$ in state 0

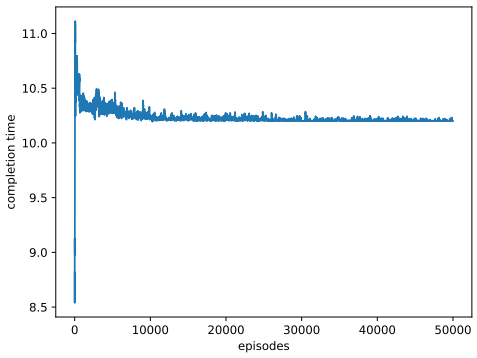


(f) Policy at the client note $R1$ in state 1

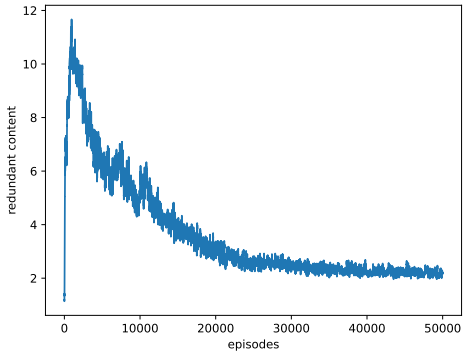
Figure A.4 – Results with REINFORCE algorithm



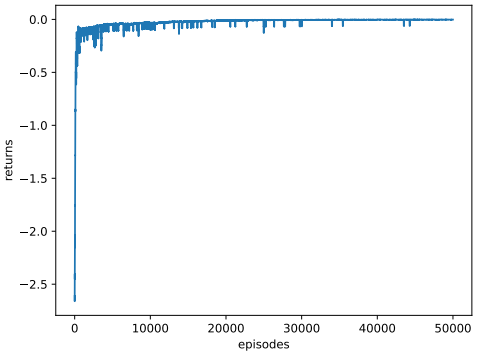
(a) Rank C_{innov} at the client node $R1$



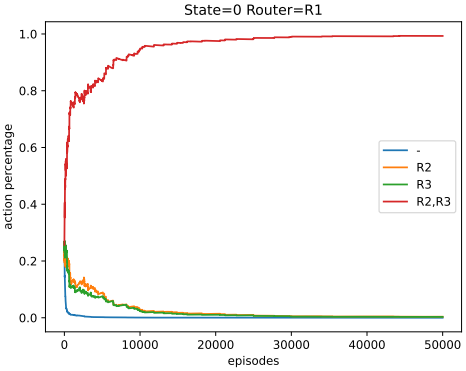
(b) Completion time T_{innov} of content retrieval at the client node



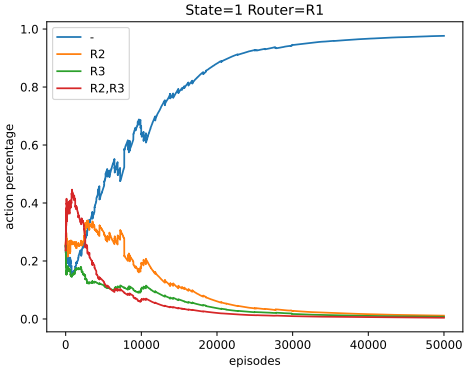
(c) Redundant content C_{red}



(d) Returns G



(e) Policy at the client note $R1$ in state 0



(f) Policy at the client note $R1$ in state 1

Figure A.5 – Results with REINFORCE algorithm with baseline

of the node is 2, and the content of size 20 is received in around 10 time units plus some propagation delay. The conservative policy in state 1 helps reduce the redundant content flowing in the network. The policy in our case converges to a deterministic policy, because the problem accepts a deterministic policy.

Figure A.5 presents the results of the REINFORCE algorithm with baseline. An estimated state-value function is used as a baseline. Figure A.5 represents results for REINFORCE with baseline. Figures A.5e and A.5f present the policy in the two states. Comparing Figures A.5e, A.5e and Figures A.4f, A.5f, it seems clear that REINFORCE with baseline learns faster and the policy converges faster to an optimal policy.

A.3.3 Deep Reinforcement Learning with Policy Methods

We get some good results with the RL techniques; however, challenges may arise when attempting to apply them to a real, more complex problem, *e.g.,* a complex network with larger content to retrieve. A common obstacle encountered with RL techniques is the dimension of the solution space. The size of the solution space of the problem grows exponentially with each additional feature describing the state [114]. This problem is described by Bellman [115] as *the curse of dimensionality.*

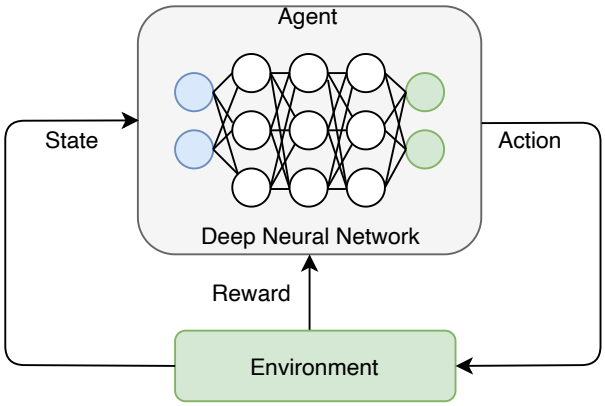
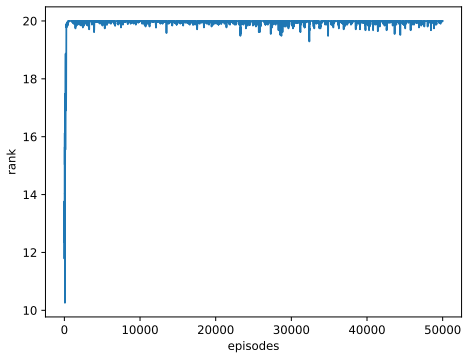
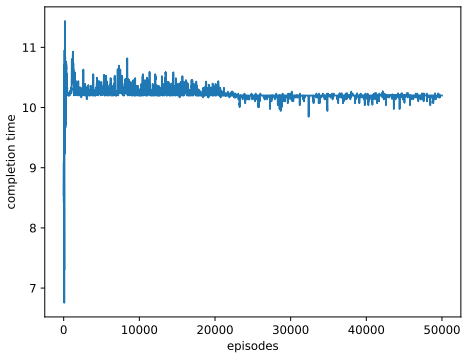


Figure A.6 – Deep Reinforcement Learning model

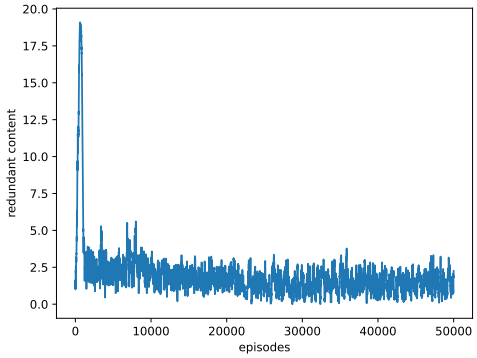
Considering the complexity of our problem in a more practical network model, we explored Deep Reinforcement Learning (DRL) approaches. DRL combines deep neural networks and reinforcement techniques. RL allows an agent to learn by trial and error to reach a particular goal with the training data set entirely dependent on the policy. DRL uses a neural network to represent a parametrized state-value function or a policy where parameters are the weights. This incorporation allows agents to use extensive unstructured input data to decide on the optimal actions. It is suitable for problems with large state space. DRL is nowadays being used for many applications, varying from video games, computer vision, robotics, healthcare, finance, and many more [116].



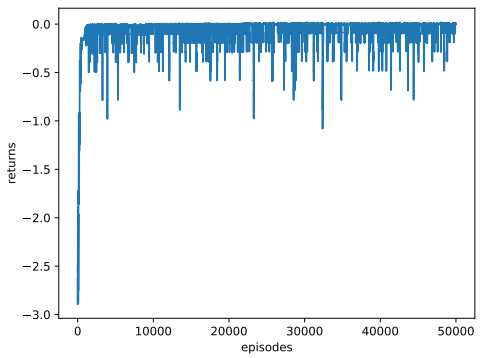
(a) Rank C_{innov} at the client node $R1$



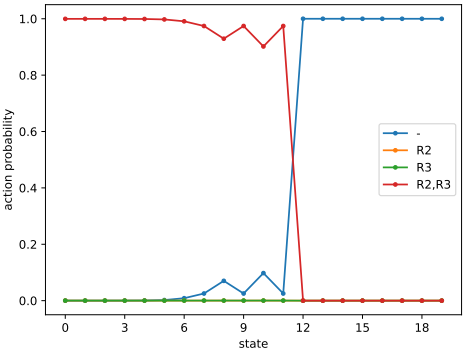
(b) Completion time T_{innov} of content retrieval at the client node



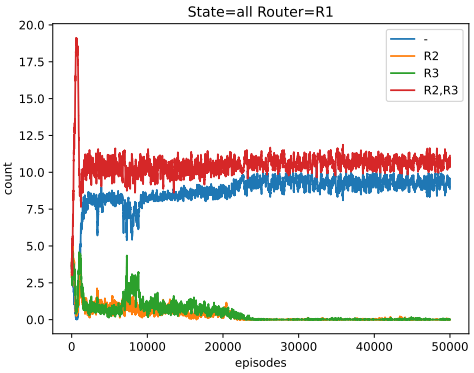
(c) Redundant content C_{red}



(d) Returns G



(e) Policy at the client note $R1$ in for all states



(f) Number of occurrences of actions per episode

Figure A.7 – Results with Deep RL

The Proximal Policy Optimization algorithm (PPO) [117] from *stable baselines* [118], was used in our Python simulator. The system parameters included the same action space as the REINFORCE algorithm. The state represents the progress of a node in terms of content retrieved for a generation, *i.e.,* the rank. Since DRL can work well with large state/observation space, simplifying state space is not required; we use all 21 states. The neural network model takes the state observation as input and gives the policy as output.

Figure A.7a presents the results for the considered DRL algorithm. Figures A.7a, A.7b and A.7c indicate that DRL performs well in terms of retrieval of complete content, completion time, and redundant content. Figure A.7e presents the policy for the last episode. The policy takes the action of sending on all faces in the initial states and switches to action zero at the end of content retrieval to avoid redundant content. Note that state 20 is the terminal state, and no action is taken in this state since the client node stops sending Interests after getting the complete generation. Since presenting the evolution of policy for 21 states over the episodes is not feasible, Figure A.7f shows the frequency of actions taken in each episode. Note that the action of sending on two faces and action zero are the preferred actions. Around 60% of the time, the Interests are sent on both faces, and the remaining 40% of the time the Interests are not forwarded anywhere. Compared to REINFORCE, DRL can use a large set of observations and tends to converge faster toward the optimum behavior.

A.4 Conclusion

This appendix presented some preliminary results with different RL algorithms implemented on MICN. These results indicate that RL algorithms can provide good solutions to implement an adaptive forwarding strategy. The stochastic policy ensures that the main goals of MICN are achieved while reducing the side effect of redundant traffic observed with MICN and other NC algorithms. These results provide a good base for further exploring the integration of RL techniques in MICN to improve its performance.

B

Synthèse

B.1 Contexte

La distribution de contenu est l'une des principales utilisations de l'Internet d'aujourd'hui, et la majeure partie du trafic de l'Internet est constituée de contenu vidéo et multimédia [10]. L'intérêt croissant pour ce type de contenu, couplée à l'évolution des comportements des consommateurs et à l'augmentation de la quantité de données échangées (à partir d'ordinateurs, de portables, de mobiles, de systèmes intelligents, de maisons connectées, de dispositifs IoT, etc...) ont entraîné un changement de paradigme. Des propositions de nouvelles architectures Internet plus modernes et plus efficaces sont actuellement explorées pour partager efficacement les contenus.

Les réseaux centrés sur l'information (Information Centric Networking, ICN) ont été proposés pour transformer les paradigmes de l'Internet et s'acquitter de la nécessité préciser l'emplacement du contenu pour y accéder. De fait, dans leurs requêtes de contenu, les réseaux ICN suppriment l'emplacement du contenu de la liste des attributs indiqués, ce qui permet aussi de le mettre en cache n'importe ou dans le réseau, de façon transparente. C'est rendu possible par le fait que les réseaux ICN identifient le contenu sur la base d'un nom plutôt que d'un emplacement comme une adresse IP et sont conçus pour acheminer le contenu sur la base de ce nom.

Certaines architectures ICN permettent aux clients de demander du contenu par l'intermédiaire de paquets d'« intérêt » indiquant son nom. L'intérêt est relayé dans le réseau jusqu'à ce qu'il atteigne un cache ou un serveur qui possède une copie du contenu demandé. Le contenu est alors envoyé aux clients à l'origine des intérêts par le chemin inverse de celui suivi par ces intérêts.

Les réseaux ICN s'affranchissent de la communication de bout en bout,

permettant donc naturellement l'utilisation simultanés d'interfaces multiples et facilitent la mise en cache dans le réseau. Ils permettent aussi une gestion transparente de la mobilité des clients car aucune information d'emplacement n'est explicitement indiquée et parce que la communication ne se limite pas à la communication entre deux points. Une requête pour du contenu peut être satisfaite par une copie proche déjà présente dans le cache d'un routeur. Chaque contenu dans un réseau ICN peut être signé et chiffré individuellement, contrairement à la communication traditionnelle qui chiffre uniquement les échanges dans un canal entre deux points. Ceci permet aussi une meilleure prise en charge de la mobilité.

L'objectif principal des réseaux ICN est de faciliter l'accès au contenu en utilisant explicitement l'aspect « orienté contenu » dans le format des demandes des clients et dans l'architecture du réseau. Néanmoins, les réseaux ICN ne résolvent pas instantanément tous les problèmes de communication et, dans certains cas, leur fonctionnement peut être amélioré ou optimisé. Lorsque plusieurs clients sont intéressés par un contenu de volume important, les réseaux ICN sont confrontés à des problèmes qui peuvent dégrader leurs performances, notamment des goulets d'étranglement, des limitations en bande passante et des pertes de paquets. En outre, l'utilisation optimale et simultanée de plusieurs chemins disponibles reste difficile avec les réseaux ICN classiques, en particulier pour des scénarios avec de multiples clients et de multiples sources.

Il existe un équilibre entre les avantages naturels des réseaux ICN et les limites rencontrées dans certains scénarios. D'une part, les réseaux ICN prennent intrinsèquement en charge la transmission de contenu par chemins multiples, et il a été observé qu'ils permettent une augmentation du débit [4]. En effet, les clients peuvent récupérer le contenu plus rapidement en exploitant la bande passante de plusieurs interfaces en même temps. D'autre part, la diffusion fiable du contenu reste un défi en particulier quand un haut niveau de performance est demandé. De fait, le débit optimal ne peut être atteint dans les scénarios de multidiffusion que si le contenu est transmis sur un ensemble optimal d'arbres de multidiffusion [5]. Or la recherche d'arbres de multidiffusion optimaux dans les réseaux à grande échelle pose des problèmes en cas d'évolution dynamique de la topologie, notamment en cas de défaillances ou de disparition de liens. Elle est également compliquée dans les réseaux ICN en raison de leur infrastructure distribuée avec une évolution dynamique et indépendante et de la mise en cache du contenu, alors qu'une connaissance complète du réseau et une coordination entre les multiples entités et routeurs du réseau seraient nécessaires pour optimiser les communications.

Le codage réseau a été proposé pour résoudre certains problèmes des réseaux ICN lors de la récupération de contenus de gros volumes [6]. L'idée principale du codage réseau est de permettre aux nœuds du réseau de « mélanger » les paquets dans à l'intérieur du réseau (via des combinaisons linéaires par exemple), contrairement au routage, où les paquets sont seulement relayés tels quels.

Le contenu est divisé en « segments » et des segments codés sont généré en combinant les segments sources (initiaux) des contenus. Les nœuds du réseau, y compris les sources et les routeurs intermédiaires, peuvent effectuer des opérations de codage réseau sur les segments du contenu et renvoyer des segments codés à l'intérieur de paquets de données ICN. Les paquets de données contiennent

également les vecteurs d'encodage des segments codés qui fournissent des informations sur le codage, c'est-à-dire les coefficients de la combinaison linéaire que le segment codé représente.

L'association du codage réseau au concept de réseaux ICN apporte un ensemble d'avantages. Une requête de segments codés d'un contenu peut être satisfaite par n'importe quel segment codé au lieu d'un seul segment spécifique, ce qui réduit la granularité d'une requête pour du contenu. Le codage du réseau permet d'envoyer les demandes des clients sur plusieurs interfaces sans avoir à les transmettre sur un ensemble optimal d'arbres de multidiffusion. Le mélange (recodage) des segments à l'intérieur du réseau augmente la diversité du contenu présent dans le réseau. Cette diversité accrue du contenu augmente le taux de réussite des caches. Le codage réseau permet à aux réseaux ICN d'utiliser efficacement plusieurs chemins et, par conséquent, d'augmenter les débits dans les applications impliquant des données volumineuses. Outre les gains de débit, le codage réseau apporte des avantages tels que la résilience aux pertes.

Un client qui demande un contenu codé ne pourra décoder le contenu original que lorsqu'il aura reçu suffisamment des paquets codés qui soient linéairement indépendants. Les paquets codés linéairement indépendants sont donc nécessaires pour que les clients puissent effectuer des opérations de décodage. Cependant, justement, il n'est pas simple de s'assurer que le contenu codé retourné aux clients soit linéairement indépendant de ce que chacun a déjà collecté. En réalité, comme les réseaux ICN conservent des segments codés dans les caches des routeurs, à l'intérieur du réseau lui-même, et du fait que tout segment codé puisse répondre à un intérêt, il y a une forte probabilité que le client reçoive à nouveau le même segment codé de nœuds voisins, après les réponses à ses premiers intérêts. Plusieurs solutions ont été proposées dans la littérature pour assurer la récupération d'un ensemble complet donc décodable de segments codés linérairement indépendants, y compris l'envoi dans chaque intérêt de l'ensemble des vecteurs d'encodage de tous les paquets codés déjà reçus. Les propositions précédemment publiées présentent néanmoins certaines faiblesses en matière d'équilibre entre le coût de la récupération d'un ensemble complet de contenu et la récupération du contenu avec le débit le plus grand possible. Une question ouverte est de savoir comment les améliorer pour tirer pleinement parti du codage réseau dans les réseaux ICN.

B.2 Objectifs de la thèse

Motivé par les avantages que le codage réseau peut apporter aux réseaux ICN en matière de débit, l'objectif principal de cette thèse est de proposer un protocole qui intègre efficacement le codage réseau et les principes des réseaux ICN pour surmonter les défauts de travaux précédents.

Nous choisissons « Named Data Networking (NDN) » [29], une architecture ICN basée sur les intérêts, pour intégrer le codage réseau aux réseaux ICN. L'objectif principal du protocole est de garantir que les clients récupèrent un ensemble décodable de segments codés avec le débit maximal possible. Pour récupérer un ensemble décodable de contenu codé, il est impératif de concevoir un « schéma de nommage » approprié. Le schéma de nommage comprend les noms donnés aux segments codés du contenu dans les paquets ICN et les méta-

informations associées dans les champs d'options. Il doit permettre aux intérêts d'informer les routeurs du réseau de ce dont le client à besoin pour décoder le contenu. Notre objectif est de concevoir un schéma de nommage pour codage réseau qui tienne compte des principes de NDN et permette le traitement de plusieurs intérêts en parallèle dans le réseau.

Le deuxième objectif dans la conception du protocole est d'assurer un débit maximal dans les scénarios de transfert de contenus volumineux. Cet objectif se traduit par une stratégie de relayage qui garantit que suffisamment d'intérêts sont propagés dans le réseau.

Le troisième objectif de cette thèse est d'optimiser l'utilisation des ressources du réseau tout en assurant un débit maximal. L'utilisation des ressources du réseau est améliorée en optimisant le nombre de paquets d'intérêt et de données envoyés dans le réseau. Cette optimisation permet de s'assurer que le protocole atteint son objectif principal sans inonder inutilement le réseau.

L'objectif final de la thèse est une étude sur la généralisation des mécanismes introduits pour atteindre l'objectif précédent : la construction de sous-ensembles d'espaces vectoriels tels que lorsqu'un vecteur est arbitrairement pris de chaque sous-ensemble, l'ensemble de vecteurs résultants est linéairement indépendant. Il s'agit notamment d'explorer d'autres familles qui satisfont à cette propriété et de vérifier si elles conviennent ou non au codage réseau.

B.3 Résumé des Contributions

Afin de concevoir une architecture intégrant le codage réseau et les réseaux NDN, nous commençons par étudier les travaux précédents. Nous analysons en détail les choix de conception des architectures intégrant le codage réseau et les réseaux ICN dans la littérature.

La première contribution de cette thèse, présentée dans le chapitre 3, fournit une solution pour la compression dans les intérêts des informations décrivant l'ensemble des segments codés reçus en utilisant une approche basée sur les codes linéaires. De nombreux travaux proposés dans le passé ajoutent les informations du contenu reçues dans les intérêts pour s'assurer que le client reçoive un ensemble décodable de segments codés. Cependant, cette information n'est pas ajoutée de manière suffisamment compacte et entraîne un surcoût important dans les paquets d'intérêt. Notre solution permet de réduire cette surcharge.

La deuxième contribution de cette thèse est également présentée dans le chapitre 3. Nous présentons une analyse formelle des stratégies de relayage des intérêts avec NetCodCCN [68]. NetCodCCN représente une famille de protocoles qui vise à atteindre des objectifs similaires aux nôtres. Notre analyse formelle montre que les stratégies de relayage de ces protocoles ne sont pas toujours efficaces et qu'il est important de fournir des preuves de protocole.

La principale contribution de cette thèse est l'architecture MICN qui intègre le codage réseau aux réseaux NDN, et elle est présentée dans le chapitre 4. Nous proposons un schéma de nommage spécifique appelé MILIC (Multiple Interests for Linearly Independent Content). MILIC ajoute un index dans le nom présent dans les intérêts. L'ensemble de ces indices représente des sous-ensembles prédéfinis de vecteurs d'encodage (suivant un ensemble de contraintes mathématiques). Le segment codé en réponse à un intérêt avec un index MILIC ne

peut avoir qu'un vecteur d'encodage appartenant à un sous-ensemble prédéfini. Les sous-ensembles garantissent que le segment codé correspondant à chaque indice est linéairement indépendant de l'ensemble des autres. Nous définissons ces sous-ensembles et fournissons des preuves formelles que l'indépendance linéaire du contenu pour différents sous-ensembles est assurée. MICN assure que chaque client recevra un ensemble complet et décodable de segments codés avec un débit maximal.

Le débit maximal de MICN est obtenu au prix d'une inondation du réseau avec des paquets d'intérêts. Une conséquence de cette inondation est la génération de trafics redondant dans le réseau. La contribution suivante de cette thèse est de proposer des stratégies dynamiques de relayage des intérêts dans le chapitre 5 pour réduire l'inondation des intérêts et par conséquent réduire le surcroît de trafic de données.

La dernière contribution de cette thèse est l'étude des familles de sous-ensembles similaire à MILIC. Nous formalisons le problème en la recherche de sous-ensembles tels qu'en choissant arbitrairement un élément dans chacun d'entre eux, l'ensemble de vecteurs résultants est linéairement indépendant. Nous étudions quelques familles générales et prouvons qu'elles sont équivalentes à MILIC. Nous proposons également quelques exemples de constructions et solutions algébriques alternatives.

B.4 Plan de la thèse

La thèse est organisée comme suit :

Le chapitre 2 donne un aperçu des propriétés fondamentales des réseaux ICN en général et des réseaux NDN en particulier. Il présente le contexte et une vue d'ensemble des travaux existants réalisés dans le domaine des réseaux NDN et du codage réseau.

Le chapitre 3 analyse les choix de conception effectués dans certains travaux préexistants avec des hypothèses similaires à celles qui nous intéressent.

Le chapitre 4 présente MICN, une architecture qui intègre le codage réseau aux réseaux NDN afin d'obtenir un débit maximal pour le transfert de données volumineuses. Le chapitre présente une architecture détaillée de MICN, les résultats et les avantages de MICN par rapport à NDN et à l'une des solutions préexistantes. En outre, le chapitre observe également l'existence de l'inondation des intérêts comme un effet secondaire du débit maximum que MICN permet d'atteindre, et propose un mécanisme initial pour y remédier en l'atténuant.

Le chapitre 5 présente quelques techniques de relayage d'intérêts modifiées pour remédier au problème d'inondation observé dans le chapitre 4.

Le chapitre 6 présente la recherche de sous-ensembles dont les transversaux sont linéairement indépendants comme un problème mathématique formel appelés SELIT. Nous prouvons qu'un grand nombre de familles de

constructions qui sont des solutions du problème SELIT sont équivalentes à notre construction MILIC (introduite dans le chapitre 4).

Le chapitre 7 conclut cette thèse et présente quelques perspectives futures et des directions prometteuses pour les architectures NDN avec le codage réseau et pour des variantes de MICN.

Certaines parties, idées et figures présentées dans cette thèse ont déjà été publiées dans les publications suivantes ;

Articles de journaux:

(J1) Hirah Malik, Cédric Adjih, Claudio Weidmann, and Michel Kieffer. MICN: a Network Coding Protocol for ICN with Multiple Distinct Interests per Generation. *Computer Networks*, 187:107816, 2021.[7]

Articles de conférences et congrès

(C2) Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. Analysis of the Properties of NetcodICN Protocols. In *CORES 2020 – 5ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Lyon, France, September 2020.[8]

(C3) Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. On the Problem of Finding "Sets Ensuring Linearly Independent Transversals"(SELIT), and its Application to Network Coding. In *9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, pages 1–6. IEEE, 2020.[9]

Bibliography

[1] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D. Thornton, Diana K. Smetters, Beichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named Data Networking (NDN) Project. *NDN, Technical Report NDN-0001*, oct 2010.

[2] Alan Ford, Costin Raiciu, Mark J. Handley, and Olivier Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, January 2013.

[3] Ali Dabirmoghaddam, Maziar Mirzazad Barijough, and Jose JoaquinJ Garcia-Luna-Aceves. Understanding Optimal Caching and Opportunistic Caching at "The Edge" of Information-Centric Networks. In *Proc. 1st ACM conference on information-centric networking*, pages 47–56, 2014.

[4] Giuseppe Rossini and Dario Rossi. Evaluating CCN multi-path interest forwarding strategies. *Computer Communications*, 36(7):771–778, 2013.

[5] Yunnan Wu, Philip A. Chou, and Kamal Jain. A Comparison of Network Coding and Tree Packing. In *Proc. 2004 IEEE International Symposium on Information Theory (ISIT)*, page 143, 2004.

[6] Marie-Jose Montpetit, Cedric Westphal, and Dirk Trossen. Network Coding Meets Information-Centric Networking: An Architectural Case for Information Dispersion Through Native Network Coding. In *Proc. 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 31–36, 2012.

[7] Hirah Malik, Cédric Adjih, Claudio Weidmann, and Michel Kieffer. MICN: a network coding protocol for ICN with multiple distinct interests per generation. *Computer Networks*, 187:107816, 2021.

[8] Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. Analysis of the properties of NetcodICN protocols. In *CORES 2020 – 5ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Lyon, France, September 2020.

[9] Hirah Malik, Cédric Adjih, Michel Kieffer, and Claudio Weidmann. On the Problem of Finding "Sets Ensuring Linearly Independent Transversals" (SELIT), and its Application to Network Coding. In *Proc. 9th IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, pages 1–6, 2020.

[10] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update2017-2022. *Cisco White Paper*, 2019.

[11] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network Information Flow. *IEEE Trans. Information Theory*, 46(4):1204–1216, 2000.

[12] Vinton Cerf and Robert Kahn. A Protocol for Packet Network Intercommunication. *IEEE Trans. Communications*, 22(5):637–648, 1974.

[13] P. V. Mockapetris. Domain Names-Concepts and Facilities. RFC 1034, November 1987.

[14] P. V. Mockapetris. Domain Names-Implementation and Specification. RFC 1035, November 1983.

[15] Roy Fielding and Julian Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231, June 2014.

[16] M. Thomson E. Damaggio and B. Raymor. Generic Event Delivery Using HTTP Push. RFC 8030, December 2016.

[17] Cisco Annual Internet Report (2018-2023). *Cisco White Paper*, 2020.

[18] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[19] Al-Mukaddim Khan Pathan, Rajkumar Buyya, et al. A Taxonomy and Survey of Content Delivery Networks . *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 4:70, 2007.

[20] Charles Severance. Van Jacobson: Content-Centric Networking. *IEEE Computer*, 46(1):11–13, 2013.

[21] George Xylomenos, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049, 2013.

[22] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. A Survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[23] Kostas Pentikousis, Borje Ohlman, Daniel Corujo, Gennaro Boggia, Gareth Tyson, E. Davies, Antonella Molinaro, and Suyong Eum. Information-Centric Networking: Baseline Scenarios. RFC 7476, March 2015.

[24] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proc. ACM SIGCOMM 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 181–192, 2007.

[25] Bengt Ahlgren, Matteo D'Ambrosio, Marco Marchisio, Ian Marsh, Christian Dannewitz, Börje Ohlman, Kostas Pentikousis, Ove Strandberg, René Rembarz, and Vinicio Vercellone. Design Considerations for a Network of Information. In *Proc. 2008 ACM CoNEXT Conference*, pages 1–6, 2008.

[26] Nikos Fotiou, Dirk Trossen, and George C. Polyzos. Illustrating a publish-subscribe Internet architecture. *Telecommunication Systems*, 51(4):233–245, 2012.

[27] Dirk Trossen and George Parisis. Designing and Realizing an Information-Centric Internet. *IEEE Communications Magazine*, 50(7):60–67, 2012.

[28] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking Named Content. In *Proc. 5th ACM International Conference on Emerging Networking Experiments and Technologies*, pages 1–12, 2009.

[29] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[30] Marc Mosko, Ignacio Solis, and Christopher A. Wood. Content-Centric Networking (CCNx) Semantics. RFC 8569, 2019.

[31] Kostas Pentikousis, Borje Ohlam, Elwin Davies, Spiros Spirou, and Gennaro boggia. Information-Centric Networking: Evaluation and Security Considerations. RFC 7945, 2016.

[32] Dirk Kutscher, Suyong Eum, Kostas Pentikousis, Ioannis Psaras, Daniel Corujo, Damien Saucez, Thomas C. Schmidt, and Matthias Waehlisch. Information-Centric Networking (ICN) Research Challenges. RFC 7927, 2016.

[33] Design Choices and Differences for NDN and CCNx 1.0 Implementations of Information-Centric Networking. Internet-Draft draft-icnrg-harmonization-00, ICNRG, 2017.

[34] Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham,

Steve Dibenedetto, Chengyu Fan, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, and Lan Wang. NFD Developer's Guide. *NDN, Technical Report NDN-0021*, Rev. 11, aug 2021.

[35] NDN Packet Format Specification 0.3 documentation. https://named-data.net/doc/NDN-packet-spec/current/, 2019.

[36] Peter Gusev and Jeff Burke. NDN-RTC: Real-Time Videoconferencing over Named Data Networking. In *Proc. 2nd ACM Conference on Information-Centric Networking*, pages 117–126, 2015.

[37] Reaz Ahmed, Md Faizul Bari, Shihabur Rahman Chowdhury, Md Golam Rabbani, Raouf Boutaba, and Bertrand Mathieu. αRoute: Routing on Names. *IEEE/ACM Trans. on Networking*, 24(5):3070–3083, 2016.

[38] Daniel Posch, Hermann Hellwagner, and Benjamin Rainer. SAF: Stochastic Adaptive Forwarding in Named Data Networking. *IEEE/ACM Trans. on Networking*, 25(2):1089–1102, 2017.

[39] Ibrahim Abdullahi, Suki Arif, and Suhaidi Hassan. Survey on caching approaches in Information Centric Networking. *Journal of Network and Computer Applications*, 56:48–59, 2015.

[40] Ming Li, Andrey Lukyanenko, Zhonghong Ou, Antti Ylä-Jääski, Sasu Tarkoma, Matthieu Coudron, and Stefano Secci. Multipath Transmission for the Internet: A Survey. *IEEE Communications Surveys & Tutorials*, 18(4):2887–2925, 2016.

[41] Andre Gomes and Torsten Braun. Load balancing in LTE mobile networks with Information-Centric Networking. In *Proc. 2015 IEEE international conference on communication workshop (ICCW)*, pages 1845–1851, 2015.

[42] Amin Shokrollahi. Raptor Codes. *IEEE Trans. Information Theory*, 52(6):2551–2567, 2006.

[43] Carlos Anastasiades, Nikolaos Thomos, Alexander Striffeler, and Torsten Braun. RC-NDN: Raptor Codes Enabled Named Data Networking. In *Proc. 2015 IEEE International Conference on Communications (ICC)*, pages 3026–3032, 2015.

[44] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear Network Coding. *IEEE Trans. Information Theory*, 49(2):371–381, 2003.

[45] Christos Gkantsidis and Pablo Rodriguez Rodriguez. Network Coding for Large Scale Content Distribution. In *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2235–2245, 2005.

[46] Philip A. Chou and Yunnan Wu. Network Coding for the Internet and Wireless Networks. *IEEE Signal Processing Magazine*, 24(5):77–85, 2007.

[47] Riccardo Bassoli, Hugo Marques, Jonathan Rodriguez, Kenneth W. Shum, and Rahim Tafazolli. Network Coding Theory: A Survey. *IEEE Communications Surveys & Tutorials*, 15(4):1950–1978, 2013.

[48] Christina Fragouli and Emina Soljanin. *Network Coding Applications*. now Publishers Inc, 2007.

[49] Christina Fragouli, Jörg Widmer, and J-Y Le Boudec. On the Benefits of Network Coding for Wireless Applications. In *Proc. 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 1–6, 2006.

[50] Tracey Ho, Ralf Koetter, Muriel Medard, David R. Karger, and Michelle Effros. The Benefits of Coding over Routing in a Randomized Setting. In *Proc. 2003 IEEE International Symposium on Information Theory (ISIT)*, 2003.

[51] Tracey Ho, Muriel Medard, Jun Shi, Michelle Effros, and David R. Karger. On Randomized Network Coding. In *Proc. 41st Annual Allerton Conference on Communication Control and Computing*, pages 11–20, 2003.

[52] Dirk Trossen, Mikko Sarela, and Karen Sollins. Arguments for an Information-Centric Internetworking Architecture. *ACM SIGCOMM Computer Communication Review*, 40(2):26–33, 2010.

[53] Abinesh Ramakrishnan, Cedric Westphal, and Jonnahtan Saltarin. Adaptive Video Streaming over CCN with Network Coding for Seamless Mobility. In *Proc. 2016 IEEE International Symposium on Multimedia (ISM)*, pages 238–242, dec 2016.

[54] Vincent Roca and Belkacem Teibi. Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME (RFC 8681), January 2020. RFC 8681, Standards Track, TSVWG (Transport Area) working group of IETF (Internet Engineering Task Force), https://www.rfc-editor.org/rfc/rfc8681.html.

[55] Kazuhisa Matsuzono, Hitoshi Asaeda, and Cedric Westphal. Network Coding for Content-Centric Networking / Named Data Networking: Requirements and Challenges. Internet-Draft draft-irtf-nwcrg-nwc-ccn-reqs-06, Internet Engineering Task Force, July 2021. Work in progress.

[56] Philip A. Chou, Yunnan Wu, and Kamal Jain. Practical Network Coding. In *Proc. 41st Annual Allerton Conference on Communication Control and Computing*, number 1, pages 40–49, 2003.

[57] Jay Kumar Sundararajan, Devavrat Shah, Muriel Médard, Szymon Jakubczak, Michael Mitzenmacher, and Joao Barros. Network Coding Meets TCP: Theory and Implementation. *Proceedings of the IEEE*, 99(3):490–512, 2011.

[58] Qinghua Wu, Zhenyu Li, and Gaogang Xie. CodingCache: Multipath-aware CCN Cache with Network Coding. In *Proc. 3rd ACM SIGCOMM Workshop on Information-centric Networking*, pages 41–42, 2013.

[59] Qinghua Wu, Zhenyu Li, Gareth Tyson, Steve Uhlig, Mohamed Ali Kaafar, and Gaogang Xie. Privacy-Aware Multipath Video Caching for Content-Centric Networks. *IEEE Journal on Selected Areas in Communications*, 34(8):2219–2230, 2016.

[60] Kai Lei, Shangru Zhong, Fangxing Zhu, Kuai Xu, and Haijun Zhang. A NDN IoT Content Distribution Model with Network Coding Enhanced Forwarding Strategy for 5G. *IEEE Trans. on Industrial Informatics*, 14(6):2725–2735, 2017.

[61] Jaime Llorca, Antonia M. Tulino, Kyle Guan, and Daniel C Kilper. Network-Coded Caching-Aided Multicast for Efficient Content Delivery. In *Proc. 2013 IEEE International Conference on Communications (ICC)*, pages 3557–3562, 2013.

[62] Yan Liu and Shun Zheng Yu. Network coding-based multisource content delivery in Content Centric Networking. *Journal of Network and Computer Applications*, 64:167–175, 2016.

[63] Guoqiang Zhang and Ziqu Xu. Combining CCN with network coding: An architectural perspective. *Computer Networks*, 94:219–230, January 2016.

[64] Wai Xi Liu, Shun Zheng Yu, Guang Tan, and Jun Cai. Information-centric networking with built-in network coding to achieve multisource transmission at network-layer. *Computer Networks*, 115:110–128, 2017.

[65] Kazuhisa Matsuzono, Hitoshi Asaeda, and Thierry Turletti. Low Latency Low Loss Streaming using In-Network Coding and Caching. In *Proc. IEEE INFOCOM*, pages 1–9, 2017.

[66] Muhammad Bilal and Shin-Gak Kang. Network-Coding Approach for Information-Centric Networking. *IEEE Systems Journal*, 13(2):1376–1385, 2018.

[67] Nidhi Lal, Shishupal Kumar, and Vijay Kumar Chaurasiya. A Network-Coded Caching-Based Multicasting Scheme for Information-Centric Networking (ICN). *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 43(3):427–438, 2019.

[68] Jonnahtan Saltarin, Eirina Bourtsoulatze, Nikolaos Thomos, and Torsten Braun. NetCodCCN: A Network Coding approach for Content-Centric Networks. In *Proc. of IEEE INFOCOM*, pages 1–9, 2016.

[69] Jonnahtan Saltarin, Eirina Bourtsoulatze, Nikolaos Thomos, and Torsten Braun. Adaptive Video Streaming With Network Coding Enabled Named Data Networking. *IEEE Trans. Multimedia*, 19(10):2182–2196, 2017.

[70] Eirina Bourtsoulatze, Nikolaos Thomos, Jonnahtan Saltarin, and Torsten Braun. Content-Aware Delivery of Scalable Video in Network Coding Enabled Named Data Networks. *IEEE Trans. Multimedia*, 20(6):1561–1575, 2017.

[71] Nikolaos Thomos, Eymen Kurdoglu, Pascal Frossard, and Mihaela Van der Schaar. Adaptive Prioritized Random Linear Coding and Scheduling for Layered Data Delivery From Multiple Servers. *IEEE Trans. Multimedia*, 17(6):893–906, 2015.

[72] Burton H Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors . *Communications of the ACM*, 13(7):422–426, 1970.

[73] Jonnahtan Eduardo Saltarin De Arco. *Network Coding Enabled Named Data Networking Architectures*. PhD thesis, University of Bern, 2017.

[74] Jacobus Hendricus Lint. *Introduction to Coding Theory*. Springer, 1992.

[75] Jonnahtan Saltarin, Eirina Bourtsoulatze, Nikolaos Thomos, and Torsten Braun. NetCodCCN: a Network Coding approach for Content-Centric Networks. In *Proc. IEEE INFOCOM 2016*, 2016.

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

[77] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michalle Effros, Jun Shi, and Ben Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Information Theory*, 52(10):4413–4430, 2006.

[78] PlanetLab. https://www.planet-lab.org/.

[79] Ziv Bar-Yossef, Yitzhak Birk, T. S. Jayram, and Tomer Kol. Index Coding With Side Information. *IEEE Trans. Information Theory*, 57(3):1479–1494, 2011.

[80] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients. *IEEE Trans. Information Theory*, 52(6):2825–2830, 2006.

[81] Michelle Effros, Salim El Rouayheb, and Michael Langberg. An Equivalence Between Network Coding and Index Coding. *IEEE Trans. Information Theory*, 61(5):2478–2487, 2015.

[82] Y. Liu, P. Sadeghi, F. Arbabjolfaei, and Y. Kim. Capacity Theorems for Distributed Index Coding. *IEEE Trans. Information Theory*, 66(8):4653–4680, 2020.

[83] Theodore C. Bergstrom. Maximal Elements of Acyclic Relations on Compact Sets. *Journal of Economic Theory*, 10(3):403 – 404, 1975.

[84] Edward Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta mathematicae*, 1(16):386–389, 1930.

[85] Alfred Geroldinger and Imre Ruzsa. *Combinatorial Number Theory and Additive Group Theory*. Springer Science & Business Media, 2009.

[86] John R. Silvester. Determinants of block matrices. *The Mathematical Gazette*, 84(501):460–467, 2000.

[87] Jonnahtan Saltarin, Torsten Braun, Eirina Bourtsoulatze, and Nikolaos Thomos. PopNetCod: A Popularity-based Caching Policy for Network Coding enabled Named Data Networking. In *proc. 2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 271–279, 2018.

[88] Kostas Pentikousis, Borje Ohlman, Elwyn B Davies, Gennaro Boggia, and Spiros Spirou. Information-Centric Networking: Evaluation and Security Considerations. *Internet Engineering Task Force, Internet-Draft draft-irtf-icnrg-evaluation-methodology-05, Apr. 2016, work in progress*, 2016.

[89] Shweta Agrawal, Dan Boneh, Xavier Boyen, and David Mandell Freeman. Preventing Pollution Attacks in Multi-source Network Coding. In *Proc. International Workshop on Public Key Cryptography*, pages 161–176. Springer, 2010.

[90] Yingdi Yu, Alexander Afanasyev, David Clark, KC Claffy, Van Jacobson, and Lixia Zhang. Schematizing Trust in Named Data Networking. In *Proc. 2nd ACM Conference on Information-Centric Networking*, pages 177–186, 2015.

[91] Joao P Vilela, Luísa Lima, and Joao Barros. Lightweight Security for Network Coding. In *Proc. 2008 IEEE International Conference on Communications (ICC)*, pages 1750–1754, 2008.

[92] Denis Charles, Kamal Jain, and Kristin Lauter. Signatures for Network Coding. In *Proc. 40th Annual Conference on Information Sciences and Systems*, pages 857–863, 2006.

[93] Shweta Agrawal and Dan Boneh. Homomorphic MACs: MAC-Based Integrity for Network Coding. In *Proc. International Conference on Applied Cryptography and Network Security*, pages 292–305. Springer, 2009.

[94] Ryma Boussaha, Yacine Challal, Malika Bessedik, and Abdelmadjid Bouabdallah. Towards Authenticated Network Coding for Named Data Networking. In *Proc. 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2017.

[95] Xiaoyan Hu, Shaoqi Zheng, Jian Gong, Guang Cheng, Guoqiang Zhang, and Ruidong Li. Enabling Linearly Homomorphic Signatures in Network Coding-based Named Data Networking. In *Proc. 14th International Conference on Future Internet Technologies*, pages 1–4, 2019.

[96] Dave Oran. Considerations in the Development of a QoS Architecture for CCNx-Like Information-Centric Networking Protocols. RFC 9064, 2016.

[97] Yongmao Ren, Jun Li, Shanshan Shi, Lingling Li, Guodong Wang, and Beichuan Zhang. Congestion control in named data networking: A survey. *Computer Communications*, 86:1–11, 2016.

[98] Milad Mahdian, Somaya Arianfar, Jim Gibson, and Dave Oran. MIRCC: Multipath-aware ICN Rate-based Congestion Control. In *Proc. 3rd ACM Conference on Information-Centric Networking*, pages 1–10, 2016.

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

[100] Nicolas Kuhn, Emmanuel Lochin, Francois Michel, and Michael Welzl. Coding and congestion control in transport. Internet-Draft draft-irtf-nwcrg-coding-and-congestion-04, Internet Engineering Task Force, October 2020. Work in progress.

[101] William Kautz and Roy Singleton. Nonrandom Binary Superimposed Codes. *IEEE Trans. Information Theory*, 10(4):363–377, 1964.

[102] Konstantin Avrachenkov and Peter Jacko. CCN Interest Forwarding Strategy as Multi-Armed Bandit Model with Delays . In *Proc. 2012 6th International Conference on Network Games, Control and Optimization (NetGCooP)*, pages 38–43, 2012.

[103] Ian Vilar Bastos and Igor Monteiro Moraes. A Forwarding Strategy based on Reinforcement Learning for Content-Centric Networking. In *2016 7th International Conference on the Network of the Future (NOF)*, pages 1–5. IEEE, 2016.

[104] Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio, Dario Rossi, and Giuseppe Rossini. INFORM: a dynamic INterest FORwarding Mechanism for Information Centric Networking. In *Proc. 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, pages 9–14, 2013.

[105] Bo Fu, Liang Qian, Yuting Zhu, and Liangpeng Wang. Reinforcement Learning-Based Algorithm for Efficient and Adaptive Forwarding in Named Data Networking. In *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1–6, 2017.

[106] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[107] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[108] Justin A Boyan and Michael L Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems*, pages 671–678, 1994.

[109] Amar Abane, Mehammed Daoui, Samia Bouzefrane, and Paul Muhlethaler. A Lightweight Forwarding Strategy for Named Data Networking in Low-end IoT. *Journal of Network and Computer Applications*, 148:102445, 2019.

[110] Olumide Akinwande. Interest Forwarding in Named Data Networking Using Reinforcement Learning. *Sensors*, 18(10):3354, 2018.

[111] Mingchuan Zhang, Xin Wang, Tingting Liu, Junlong Zhu, and Qingtao Wu. AFSndn: A novel adaptive forwarding strategy in named data networking based on Q-learning. *Peer-to-Peer Networking and Applications*, 13(4):1176–1184, 2020.

[112] Zoubir Mammeri. Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches. *IEEE Access*, 7:55916–55950, 2019.

[113] Nitul Dutta, Shobhit K Patel, Vadim Samusenkov, and D Vigneswaran. Deep learning inspired routing in ICN using Monte Carlo Tree Search algorithm. *Journal of Parallel and Distributed Computing*, 150:104–111, 2021.

[114] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement Learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[115] Richard E Bellman. *Dynamic Programming*. Princeton university press, 1957.

[116] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[117] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[118] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Protocoles de codage réseau efficaces pour les réseaux centrés sur l'information

Mots clés : Réseaux centrés sur l'information, Named Data Networking (NDN), Codage réseau, Stratégies de relayage d'intérêts, Réseaux centrés contenu

Résumé : Les réseaux centrés sur l'information (Information Centric Networking, ICN) ont été proposés comme une alternative aux réseaux IP traditionnels. Dans les réseaux ICN, les consommateurs demandent au réseau un contenu par son nom via des paquets « intérêt », et reçoivent des données en réponse à leurs demandes sans avoir à se soucier de l'emplacement du contenu dans le réseau.

Le codage réseau (Network Coding) a été récemment introduit dans les réseaux ICN afin d'améliorer la diffusion par l'utilisation de chemins multiples et la mise en cache du contenu sans qu'une coordination soit nécessaire. Cette thèse propose une architecture, MICN, qui intégre un codage réseau au dessus d'une implémentation ICN basée sur les intérêts : Named Data Networking (NDN). L'architecture proposée permet de résoudre certains des problèmes rencontrés par les solutions ICN avec du codage réseau présentées dans le passé. Une nouvelle construction appelée MILIC (Multiple Interests for Linearly Independent Content) est introduite. Elle impose des contraintes sur la façon dont les réponses aux intérêts sont codées, dans le but d'obtenir des contenus linéairement indépendants en réponse à des intérêts multiples. Plusieurs techniques de transport modifiées et intégrées dans le protocole MICN sont proposées afin d'optimiser l'utilisation des ressources du réseau tout en conservant un débit élevé.

MILIC construit des sous-ensembles de vecteurs à partir d'un espace vectoriel donné, tels que lorsque l'on choisit arbitrairement un vecteur de chaque sous-ensemble, les vecteurs sélectionnés sont linéairement indépendants. Cette thèse le formalise comme un problème mathématique et étudie quelques solutions alternatives et prouve qu'une large classe de solutions à ce problème est équivalente à MILIC.

Title : Efficient Network Coding Protocols for Information-Centric Networks

Keywords : Information-Centric Networking (ICN), Named Data Networking (NDN), Network Coding, Interest Forwarding Strategy

Abstract : Information-Centric Networking (ICN) has been proposed as an alternative to traditional IP-based networks. In ICN, consumers request named content via Interest packets to the network and receive data as a response to their request from anywhere in network.

Network coding has been recently introduced in ICN to improve multi-path dissemination and caching of content. This thesis proposes an architecture, MICN, that provides network coding on top of an Interest-based ICN implementation : Named Data Networking (NDN). The proposed architecture helps alleviate the issues faced by network coding-enabled ICN solutions presented in the past. A novel construction called MILIC (Multiple Interests for Linearly Independent Content) is introduced that imposes constraints on how the replies to Interests are coded, intending to get linearly independent contents in response to multiple Interests. Several modified forwarding techniques integrated into the MICN protocol are proposed to optimize the network resource utilization while keeping a high throughput.

MILIC constructs subsets of vectors from a given vector space, such that when drawing arbitrarily one vector from each subset, the selected vectors are linearly independent. This thesis considers it as a mathematical problem and studies some alternative solutions and proves that a large family of solutions to this problem are equivalent to MILIC.