Routing Named Data in Information-Centric Networks

Abdelali Kerrouche

UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE MSTIC

Thèse de doctorat

Spécialité : Informatique et Réseaux

M. ABDELALI KERROUCHE

Titre :

Routage des données dans les réseaux centrés sur les contenus

Rapporteurs :

M. Pascal Lorenz, Professeur, Université de Haute Alsace

M. Yann Labit, Professeur, Université de Toulouse

Examinateurs :

Mme. Ana Rosa Cavalli, Professeure, Télécom SudParis, France

Mme. Houda Labiod, Professeure, Télécom ParisTech , France

M. Thiago Abreu, Maître de Conférences, Université Paris-Est Créteil, France

Thèse dirigée par :

M. Abdelhamid MELLOUK, Directeur de thèse, Professeur à l'UPEC, France

M. Mustapha Reda Senouci, Co-encadrant, Maître de Conférences A à l'EMP, Algérie

Soutenue le 26 mai 2017

UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE MSTIC

DOCTOR OF SCIENCE

Specialization : COMPUTER SCIENCE

Mr. ABDELALI KERROUCHE

Topic :

Routing Named Data in Information-Centric Networks

Reviewers :

Mr. Pascal Lorenz, Professeur, University of Haute Alsace

Mr. Yann Labit, Professeur, University of Toulouse

Examiners :

Ms. Ana Rosa Cavalli, Professeur, Telecom ParisTech, France

Ms. Houda Labiod, Professeur, Telecom SudParis, France

Mr. Thiago Abreu, Associate Professor, University of Paris-Est Creteil, France

Thesis supervised by :

Mr. Abdelhamid Mellouk, Supervisor, Professor at UPEC, France

Mr. Mustapha Reda Senouci, Co-supervisor, Associate Professor at EMP, Algeria

Table of contents

1 Introduction **23**
- 1.1 Scope . . . 23
- 1.2 Problem Statement and Objectives . . . 25
- 1.3 Dissertation Contributions . . . 26
- 1.4 Dissertation Organization . . . 27

2 Information Centric Networks **29**
- 2.1 Introduction . . . 29
- 2.2 ICN Fundamental Architectures . . . 30
  - 2.2.1 DONA . . . 30
    - 2.2.1.1 Naming . . . 31
    - 2.2.1.2 Name Resolution and Data Routing . . . 31
    - 2.2.1.3 Security . . . 31
    - 2.2.1.4 Transport . . . 31
    - 2.2.1.5 Caching . . . 31
    - 2.2.1.6 Mobility . . . 32
  - 2.2.2 PURSUIT (PSIRP) . . . 32
    - 2.2.2.1 Naming . . . 32
    - 2.2.2.2 Name Resolution and Data Routing . . . 32
    - 2.2.2.3 Security . . . 33
    - 2.2.2.4 Transport . . . 34
    - 2.2.2.5 Caching . . . 34
    - 2.2.2.6 Mobility . . . 34
  - 2.2.3 NetInf . . . 34
    - 2.2.3.1 Naming . . . 35
    - 2.2.3.2 Security . . . 35
    - 2.2.3.3 Transport . . . 35
    - 2.2.3.4 Caching . . . 35
    - 2.2.3.5 Mobility . . . 36
  - 2.2.4 MobilityFirst . . . 36

2.2.4.1 Naming . . . 36
2.2.4.2 Name Resolution and Data Routing . . . 36
2.2.4.3 Security . . . 37
2.2.4.4 Caching . . . 37
2.2.4.5 Mobility . . . 38
2.2.5 Named Data Networking . . . 38
2.2.5.1 NDN node architecture . . . 39
2.2.5.2 Forwarding Process . . . 41
2.2.5.3 Naming . . . 42
2.2.5.4 Security . . . 43
2.2.5.5 Transport . . . 44
2.2.5.6 Caching . . . 45
2.2.5.7 Mobility . . . 45
2.3 Comparison between the various approaches . . . 46
2.4 Conclusion . . . 47

3 Related Work **49**
3.1 Introduction . . . 49
3.2 Traditional Routing in IP . . . 50
3.3 Routing Plane in NDN . . . 51
3.4 Forwarding Plane in NDN . . . 51
3.5 Routing and Forwarding Coordination in NDN . . . 52
3.6 Building the FIB table . . . 53
3.7 NDN Routing Protocols: a survey . . . 55
3.7.1 OSPFN: OSPF for Named Data Networking . . . 55
3.7.2 NLSR: NDN Link State Routing . . . 56
3.7.3 Hyperbolic routing . . . 59
3.8 QoS based routing in NDN network . . . 61
3.8.1 Problem Formulation . . . 61
3.9 Forwarding Strategies: a Survey . . . 62
3.9.1 High level taxonomy . . . 62
3.9.1.1 Forwarding strategies that handle the network failure . . . 62
3.9.1.2 Forwarding strategies that handle the caching issue . . . 63
3.9.1.3 Forwarding strategies that handle the security issue . . . 64
3.9.1.4 Forwarding strategies that handle the congestion issue . . . 64
3.9.1.5 Forwarding strategies with QoS . . . 65
3.9.1.6 Hybrid forwarding strategies . . . 67
3.10 Conclusion . . . 67

4 QoS-based Forwarding Strategy for NDN **69**
4.1 Introduction . . . 69
4.2 Reinforcement Learning . . . 70
4.2.1 Mathematical Model of Reinforcement Learning . . . 71
4.2.2 Value functions . . . 72

4.2.3 Rewards . . . 73
4.2.4 Definitions of functions of V-value and Q-value . . . 73
4.2.5 Value function and optimal policy . . . 73
4.2.6 Q-Learning . . . 75
4.3 Our proposal: the QoS forwarding strategy . . . 76
4.3.1 Functional Model . . . 76
4.3.2 Designing the QoS forwarding strategy . . . 78
4.3.2.1 Criterion 1: Bandwidth . . . 78
4.3.2.2 Criterion 2: Round Trip Time (RTT) . . . 81
4.3.2.3 Exploration and exploitation phases . . . 83
4.3.2.4 Forwarding decision-making algorithms . . . 85
4.4 Performance Evaluation . . . 88
4.5 Conclusion . . . 91

5 Ant Colony based QoS aware Forwarding Strategy for NDN 93
5.1 Introduction . . . 93
5.2 Ant colony optimization . . . 94
5.2.1 Meta-heuristic optimization . . . 94
5.2.2 Ant colony algorithms principles . . . 94
5.2.3 Ant system . . . 96
5.2.4 Ant Colony System (ACS) . . . 97
5.3 Our Proposal: the Ant Colony Based QoS-aware Forwarding Strategy . . . 98
5.3.1 Designing AC-QoS-FS . . . 98
5.3.1.1 Criterion 1: Bandwidth . . . 100
5.3.1.2 Criterion 2: Cost . . . 102
5.3.1.3 Criterion 3: Round Trip Time (RTT) . . . 102
5.3.1.4 The moving rule . . . 102
5.3.1.5 Pheromone update . . . 103
5.3.2 Forwarding decision-making algorithms . . . 104
5.3.2.1 After receiving an Interest . . . 104
5.3.2.2 After receiving a Data packet . . . 106
5.4 Performance Evaluation . . . 107
5.5 Conclusion . . . 108

6 Multi-Criteria QoS-based Forwarding Strategy for NDN 111
6.1 Introduction . . . 111
6.2 Our Proposal: Multi Criterion Ant Colony Based QoS-aware Forwarding Strategy(**MC-QoS-FS**) . . . 112
6.2.1 Designing **MC-QoS-FS** . . . 113
6.2.1.1 The moving rule . . . 115
6.2.1.2 Pheromone update . . . 115
6.2.2 Forwarding decision-making algorithms . . . 117
6.2.2.1 After receiving an Interest . . . 118
6.2.2.2 After receiving a Data packet . . . 119

6.3 Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (**FCR-QoS-FS**) . . . 119
6.3.1 Designing of **FCR-QoS-FS** . . . 120
6.3.1.1 Criterion 1: Bandwidth . . . 122
6.3.1.2 Criterion 2: Cost . . . 122
6.3.1.3 Criterion 3: Round Trip Time (RTT) . . . 122
6.3.1.4 Criterion 4: Interest Arrival Rate (IAR) . . . 122
6.3.1.5 Criterion 5: Data Satisfaction Rate (DSR) . . . 123
6.3.1.6 Computing Node IAR and DSR . . . 124
6.3.1.7 Updating of the IAR and DSR path . . . 124
6.3.1.8 The moving rule . . . 125
6.3.1.9 Pheromone update . . . 125
6.3.2 Forwarding decision-making algorithms . . . 126
6.3.2.1 After receiving an Interest . . . 126
6.3.2.2 After receiving a Data packet . . . 127
6.4 Performance Evaluation . . . 127
6.5 Conclusion . . . 131

7 Conclusion **135**
7.1 Summary of Contributions . . . 135
7.2 Future work . . . 137
7.3 Publications . . . 139

8 Version Française Abrégée **141**
8.1 Contexte Général de la thèse . . . 141
8.2 Problématique . . . 143
8.2.1 Routage et acheminement des paquets dans NDN . . . 144
8.2.2 Routage avec Qualité de Service . . . 145
8.2.3 Formulation du Problème . . . 145
8.3 Nos contributions . . . 146
8.3.1 QoS-FS : Stratégie d'acheminement basée sur la QoS pour NDN . . . 146
8.3.2 AC-QoS-FS : Stratégie d'acheminement avec qualité de service basée sur les colonies de fourmis . . . 148
8.3.3 MC-QoS-FS : Stratégie d'acheminement basée sur multi critères de qualité de service . . . 150
8.4 Conclusion . . . 155

Bibliographie **157**

Dedication

To my parents, my wife and my kids.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Abdelhamid MELLOUK, for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

A very special thanks go out to my co-adviser Dr.Mustapha Reda SENNOUCI, for his support and his availability.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Ana Rosa CAVALLI, Prof. Houda LABIOD, Prof. Pascal LORENZ, Prof. Yann LABIT, and Dr. Thiago ABREU, for accepting to evaluate my work and their insightful comments.

My sincere thanks also goes to Prof. Yacine AMIRAT who gave access to the laboratory and research facilities. Without his precious support it would not be possible to conduct this research.

I thank my fellow lab mates for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last three years. Also I thank my friends in the LISSI laboratory, who supported and helped me with their capabilities and valuable discussions to complete this research work.

Last but not the least, I would like to thank my family: my parents, my wife and my kids for supporting me spiritually throughout writing this thesis and my life in general.

Résumé

Les Réseaux Orientés Contenus (Information Centric Networking (ICN)) représentent un nouveau paradigme qui se développe de plus en plus dans le monde de l'Internet. Ils mettent en avant de nouvelles approches centrées sur le contenu pour concevoir une nouvelle architecture pour le réseau Internet du futur dont l'usage bascule aujourd'hui d'une communication orientée machines (hosts) vers une distribution et une récupération de contenus à large échelle.

Dans ce cadre, plusieurs architectures de type ICN ont été proposées par la communauté scientifique dans le cadre de plusieurs projets internationaux : DONA, PURSUIT, SAIL, COMET, CONVERGENCE, Named Data Networking (NDN), etc.

Nos travaux de thèse se sont focalisés sur la problématique du routage dans les réseaux de ce type, au travers d'une architecture de type NDN qui représente aujourd'hui une des architectures ICN les plus évoluées.

En particulier, nous nous sommes intéressés à concevoir et à mettre en œuvre des solutions de routage qui intègrent les métriques de qualité de service (QdS) dans les architectures NDN au regard de usages courants dans le réseau Internet. Celui-ci est en effet caractérisé par une hétérogénéité des liaisons et des conditions de trafic hautement dynamiques.

Dans ce type d'architectures, la diffusion des paquets de données est organisée en deux plans : le plan de routage et le plan d'acheminement. Ce dernier est responsable de l'acheminement des paquets sur l'ensemble des chemins disponibles au moyen d'une stratégie identifiée en amont. Le plan du routage est quant à lui utilisé uniquement pour soutenir le plan d'acheminement. De fait, les solutions que nous proposons consistent en de nouvelles stratégies d'acheminement avec QdS que nous qualifions d'adaptatives. Ces stratégies sont capables de transmettre les paquets sur de multiples chemins tout en considérant les paramètres de QdS liés à l'état du réseau et collectés en temps réel. La première approche proposée est conçue sur la base d'une méthode d'apprentissage inductif, du type Q-learning en ligne, et est utilisée pour estimer les informations collectées sur l'état dynamique du réseau.

La deuxième contribution consiste dans une stratégie d'acheminement adaptatif conçue pour les architectures NDN et prenant en compte les métriques liées à la QdS. Elle est basée sur les similarités entre le processus d'acheminement des paquets dans les architectures NDN et le comportement des fourmis lors de la recherche du plus court chemin entre leur nid et les sources de nourriture. Les techniques utilisées pour concevoir cette stratégie sont inspirées des approches d'optimisation utilisées dans les algorithmes de type « colonies de fourmis ».

Enfin, dans la dernière partie de la thèse, nous généralisons l'approche décrite ci-dessus pour l'étendre à la prise en compte simultanée de plusieurs paramètres de QdS. Sur la base de ces mêmes principes, cette approche a ensuite été étendue à la résolution des problèmes liés à la congestion.

Les résultats obtenus montrent l'efficacité des solutions proposées dans une architecture NDN et permettent ainsi de considérer les paramètres de QdS dans les mécanismes d'acheminement des

paquets ouvrant la voie à diverses applications orientées contenus sur ce type d'architecture.

Mots-clés :

Réseaux Orientés Information, Réseaux Orientés Contenus, Routage, Acheminement, Apprentissage Automatique, Apprentissage par Renforcement, Optimisation, Colonies de fourmis, Qualité de service.

Abstract

The Information Centric Networking (ICN) represents a new paradigm that is increasingly developed within the Internet world. It brings forward new content-centric based approaches, in order to design a new architecture for the future Internet, whose usage today shifts from a machine oriented communication (hosts) to a large-scale content distribution and retrieval.

In this context, several ICN architectures have been proposed by the scientific community, within several international projects: DONA, PURSUIT, SAIL, COMET, CONVERGENCE, Named Data Networking (NDN), etc.

Our thesis work has focused on the problems of routing in such networks, through a NDN architecture, which represents one of the most advanced ICN architectures nowadays.

In particular, we were interested in designing and implementing routing solutions that integrate quality-of-service metrics (QoS) in the NDN architecture in terms of current Internet usage. This latter is indeed characterized by a heterogeneity of connections and highly dynamic traffic conditions.

In this type of architecture, data packets broadcast is organized in two levels: the routing plan and the forwarding plane. The latter is responsible for routing packets on all available paths through an identified upstream strategy. The routing plan is meanwhile used only to support the forwarding plane. In fact, our solutions consist of new QoS routing strategies which we describe as adaptive. These strategies can transmit packets over multiple paths while taking into account the QoS parameters related to the state of the network and collected in real time. The first proposed approach is designed on the basis of a on-line Q-learn type inductive learning method, and is used to estimate the information collected on the dynamic state of the network.

The second contribution is an adaptive routing strategy designed for NDN architectures which considers the metrics related to QoS. It is based on the similarities between the packet forwarding process in the NDN architecture and the behavior of ants when finding the shortest path between their nest and food sources. The techniques used to design this strategy are based on optimization approaches used ”ant colonies” algorithms.

Finally, in the last part of the thesis, we generalize the approach described above to extend it to the simultaneous consideration of several QoS parameters. Based on these principles, this approach was later extended to solving problems related to congestion.

The results show the effectiveness of the proposed solutions in an NDN architecture and thus allow to consider QoS parameters in packet delivery mechanisms paving the way for various content-oriented applications on this architecture.

KeyWords:

Information-Centric Networking, Content-Centric Networking, Routing, Forwarding, Automatic Learning, Reinforcement Learning, Optimization, Ant Colonies, Quality of Service.

Acronyms

AS: Autonomous System
ACO: Ant Colony Optimization
BW: Bandwidth
CDN: Content Delivery Network
CS: Content Store
DONA: Data Oriented Network Architecture
DoS: Denial Of Service
DSR: Data Satisfaction Rate
FIB: Forwarding Information Base
GNRS: Global Name Resolution Service
GUID: Globally Unique Identifier
ICN: Information Centric-Networking
IAR: Interest Arrival Rate
NDN: Named Data Networking
NRS: Name Resolution Service
NDO: Named Data Object
NetInf: Network Information
P2P: Peer to Peer
PURSUIT: Publish Subscribe Internet Technology
PSIRP: Publish Subscribe Internet Routing Paradigm
PIT: Pending Interest Table
PLA: Packet Level Authentication
PKI: Public Key Infrastructures
PK: Public Key
QoS: Quality of Service
QoE: Quality of Experience
RTT: Round Trip Time
RL: Reinforcement Learning
RH: Resolution Handlers

Contents

Chapter 1

Introduction

Contents

1.1 Scope 23
1.2 Problem Statement and Objectives 25
1.3 Dissertation Contributions 26
1.4 Dissertation Organization 27

1.1 Scope

The Internet has known a great development since 1960s until nowadays. At the beginning it was unpredictable that the Internet would have such growth. Developed by the Unit States Department of Defense, in 1970s Internet was an academic network called ARPANET [1], which was designed to address the period needs such as sharing rare and expensive resources, such as peripherals, mainframe computers, and long-distance communication links. It was also used for the transmission of data packets between a limited number of fixed machines [2]. As result, the designed architecture has been host centric, based on a communication model which allows the connection between exactly two machines (a source and a destination).

Nowadays, the Internet has become indispensable in our daily life, since it has entered in all areas of life such as education, economics, e-commerce, etc. The number of connected devices to the Internet has reached more than fourteen billions in 2014, the number of Internet users has reached more than three billions, and billions of indexed web pages are available [3]. This tremendous growth of Internet is accompanied by a significant shift in the usage. More specifically, Internet was originally conceived to enable communication between hosts. Nowadays, it is essentially used for content distribution and retrieval, given the amount of information stored in different servers

installed around the world, causing a massive exchange across the Internet network. Furthermore, the users are in general interested in the content itself rather than its location. This shift in usage from host-centric to content-centric paradigm has motivated the emergence of many attempts to accommodate content distribution within the current Internet infrastructure. Mainly, these attempts strive to decouple contents from hosts. Earlier attempts such as peer-to-peer (P2P) networks and content delivery networks (CDN) decouples content and hosts at the application layer. Many studies show that P2P and CDN can enhance QoS and QoE of end-users; however they incur high costs and may lead to inefficient solutions.

It is noteworthy that the significant growth of the Internet and the introduction of new applications to address the emerging needs of users have leaded to the appearance of new architectural requirements such as mobility, security, ubiquity, etc. In reality, the Internet has never been designed to address these requirements. So, in order to help the evolution of Internet, various temporary solutions have begun to appear; however, most of these solution increase only the complexity of the overall architecture [1]. The architectural problems are classified as follows [1]:

- **Short-term problems**, which consist on spam, security, denial-of-service attacks and applications deployment;
- **Medium-term problems**, which consist on congestion control, Inter-domain routing, mobility, multi-homing and architectural ossification;
- **Longer-term problems**, which consist on address space depletion.

In the literature, we can find two essential approaches to resolve these challenges. The first approach is the incremental or evolutionary research, which means that an investigated system is changed incrementally and moved from one state to another supporting old potential and adding new opportunities. This can be achieved by understanding the behavior of the current Internet and identifying the existing and emerging problems, in order to solve them [4]. The second, clean-slate approach, means that every new state of system is developed from scratch, providing old possibilities on new principles [5], by designing new future Internet architecture which is better than the existing architecture, in terms of security, resilience and other proprieties without being constrained by the current Internet [4].

Many researchers believe that it is impossible to solve the architectural problems mentioned previously in the current Internet structure. Consequently, many contributions have appeared in the direction of rethinking the fundamental assumptions and design decisions, and starting from scratch. The research community nowadays intends to focus on a clean-slate approach attempts to find appropriate solution for all these issues instead on the evolutionary approach [5].

In this context, Information Centric Networking (ICN) is a new paradigm that presents an approach that has recently emerged for a new architecture of the future Internet network, centered on

content. It is inspired by the fact the Internet usage is shifting to content distribution and retrieval instead communication between hosts. The ICN approach is currently considered as a promising solution for the emergence of a new architecture of the future Internet. Several Information-centric architectures have been proposed by various projects launched around the world such as: DONA, PURSUIT, SAIL, COMET, CONVERGENCE, CCN, NDN, etc.

Among these architectures, one of them stands out clearly, it is the architecture known as NDN which is today the most advanced [6], [7], [8], [9], [10]. It is considered one of the most popular architectures for the emergence of true ICN. It was proposed by the Palo Alto Research Center (PARC) in the United States. It is one of five projects funded by the National Science Foundation (NSF) under its Future Internet Architecture program. This recent architecture is today the center of attention of the research community and several works are devoted to it.

1.2 Problem Statement and Objectives

Since the proposed architectures according to the Information Centric Networking paradigm are very recent, the researches in this area are in their beginnings. So, various research challenges still need to be addressed such as naming, security, name resolution, routing, in-network storage, caching, mobility, etc [11]. Consequently, the success of any ICN architecture dependents on the quality of solutions proposed for each one of these research challenges. In this dissertation we will investigate the routing issue in the ICN networks, through an architecture of type NDN. Consequently, our contributions will be designed and implemented according to the NDN network architecture, and also will be validated on simulation mode by using ndnSIM simulator for network simulator (ns3).

The Internet is characterized by the heterogeneity of its links and the dynamic traffic conditions. This requires taking into account the quality of service in network and especially in routing to satisfy the users requirements. This issue has been widely discussed in the literature for the IP (Internet Protocol) architecture, while it has just started in the ICN architectures.

In NDN network, the data packets delivery is also organized as in IP architecture, on two planes (routing and forwarding plane). However, the forwarding plane, by means of its forwarding strategy module is responsible of packets forwarding through available multipaths, and the routing plane is used only to help the forwarding plane. As a result, we have been interested in this dissertation in improving the quality of service perceived by the users, by designing and implementing smart forwarding strategies capable to forward the packets over multipaths while considering the QoS parameters of the real time network status. From our study three contributions have been proposed. Those they present three adaptive forwarding strategies with QoS.

1.3 Dissertation Contributions

In this dissertation, we have presented three contributions, The first contribution consists of the design and implementation of a new adaptive forwarding strategy with quality of service (QoS) of NDN, which consider only two QoS parameters (Round Trip Time (RTT) and bandwidth), called **QoS-FS**. At each node of the network, **QoS-FS** monitors, in real-time, ingoing and outgoing links of network to estimate the QoS parameters and integrate them into the different decisions taken to determine when and which interface to use to forward an incoming packet. Therefore, making forwarding decision adaptive to network conditions and preferences of user. The QoS-FS design is based on an inductive learning method, type Q-learning on-line, which we have used to estimate the collected information about the dynamic state of the network.

The second contribution also on consists the design and implementation of another adaptive forwarding strategy with QoS of NDN called Ant Colony based QoS-aware Forwarding Strategy (**AC-QoS-FS**). It is based on similarities between the NDN forwarding process and the ants behavior while searching for shortest path between their nest and a food source. The techniques used for designing **AC-QoS-FS** are borrowed from the ant colony optimization algorithm (Ant System), that was originally used to solve the famous NP-Hard Traveling Salesman Problem (TSP). **AC-QoS-FS** is able to take into account three QoS parameters (Round Tripe Time (RTT), Bandwidth and cost). In other words, we have adapted this algorithm to design **AC-QoS-FS**. Consequently, initially, the ants explore more or less randomly available paths. Then, over time, they will be adapted to the environment and the optimal paths emerge, that will be the most followed. So, at the beginning, at each node, ants are forwarded randomly to the next hop on upstream through the available interfaces. When the ant finds the requested data, it return back to the consumer node, taking the same path traversed during the research phase but in opposite direction. Furthermore, at each node, it deposits an amount of pheromone according to the real time quality of the path traversed between the food source (Producer or caching router) and the current node, in term of QoS parameters measurements (Round Trip Time, bandwidth and cost) collected by the ant. These QoS parameters measurements are updated at each node before forwarding the packet to the next hop in downstream. After that, at each node, **AC-QoS-FS** ranks the available interfaces and makes the forwarding decisions of incoming interests according the pheromone amount.

In the third contribution, we have started by generalizing the approach used in the second contribution for supporting several QoS parameters by proposing a model for designing a Multi Criterion QoS based Forwarding Strategy for NDN. After that, by applying the principles proposed for this model, we have designed and implemented a new forwarding strategy called Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (**FCR-QoS-FS**), which takes into account five QoS parameters. **FCR-QoS-FS** is designed to be able to adapt as much as possible to the network dynamic traffic, in particular to the problems of congestion. Consequently,

in addition to optimizing the QoS parameters taken into account in the second contribution (Round Trip Time, bandwidth, and cost), it takes also into account two other parameters that have a high impact on the congestion which are Interest Arrival Rate (IAR) and Data Satisfied Rate (DSR)).

1.4 Dissertation Organization

This dissertation is organized as follows. The second chapter presents the ICN paradigm for designing future Internet architecture, its principles and objectives. Furthermore it introduces an overview about the ICN fundamental architectures, proposed by the various ICN projects launched around the world. We have focused more on NDN architecture since our contributions are designed according to it. In the third chapter we present the packet delivery system organization, routing and forwarding planes, in NDN architecture. We will explain the role of each plane and the difference between them and we introduce the related literature work such as routing protocols, forwarding strategy,etc. In the fourth chapter, we presented our first contribution, which consists of a new forwarding strategy based on Quality of Service for routing in NDN (**QoS-FS**), which takes into account only two QoS parameters (RTT and bandwidth), and uses reinforcement learning as tool to estimate the QoS parameters measurements collected. In the fifth chapter, we present our second contribution, consisting on another forwarding strategy called Ant Colony based on Quality-of-Service-aware Forwarding Strategy (**AC-QoS-FS**). This latter considers three QoS parameters (RTT, bandwidth and cost), and it uses meta-heuristics Ant Colony optimization algorithm. In the sixth chapter, we present at first a designing model which is a generalization of the approach proposed in Chapter 5. Then, by applying the model principle, we propose also another forwarding strategy called Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (**FCR-QoS-FS**). The latter takes into account five QoS parameters (RTT, bandwidth, cost, Interest Arrival Rate (IAR) and Data Satisfied Rate (DSR)). Finally, we conclude this dissertation with the recapitulation of our contributions and proposed perspectives, based on the work carried out.

Chapter 2

Information Centric Networks

Contents

2.1 Introduction 29
2.2 ICN Fundamental Architectures 30
2.2.1 DONA 30
2.2.2 PURSUIT (PSIRP) 32
2.2.3 NetInf 34
2.2.4 MobilityFirst 36
2.2.5 Named Data Networking 38
2.3 Comparison between the various approaches 46
2.4 Conclusion 47

2.1 Introduction

Information-Centric Network (ICN) is a new paradigm that has recently emerged, which presents an approach centered on the contents for a new architecture of the future Internet network, instead of the current Internet architecture which is centered on hosts.

The ICN architecture dedicated for future Internet is centered on **What** while the current (IP) Internet architecture is centered on **Where**. In this latter, the hosts are named and each host has an IP address, and the names of contents are attributed by the application layer. Furthermore, the request of contents is made by the couple name of content and host address according to the conventions used. In addition, the network is responsible for connecting the two end hosts and each IP packet has two IP address that identified the source and destination hosts. However, in ICN the contents are named and the hosts names or their IP address are not necessary for retrieving the requested content. In other words, the requests of contents are made only by name independently of the

location, and network is responsible for finding the content. Several content-centric architectures have been proposed by various projects launched around the world, such as DONA, PURSUIT, SAIL, COMET, CONVERGENCE, CCN, NDN, etc. Despite the significant difference existing between the proposed architectures in composition and at operating principles, the fundamental goals are the same for all architectures, as pointed out by Van Jacobson in its Google Talk [12]. CCN goals consist on create a simple, flexible and universal communication architecture that:

- Matches to communications problems of today;
- Matches to application design models today;
- Is at least as scalable and efficient as TCP/IP;
- Is much more secure;
- Requires far less configuration.

In this chapter, we present an overview of various future Internet architectures with more focus for Named Data Networking (NDN) as our contributions are implemented and validated on the NDN architecture.

2.2 ICN Fundamental Architectures

The first architecture according to ICN paradigm was introduced in 1999 with TRIAD [2] project. Since then, several architectures have been proposed in different projects relying over the same paradigm. In this section, we give a short introduction and overview of different ICN approaches such us: DONA, PURSUIT, NDN, NetInf and MobilityFirst.

2.2.1 DONA

The Data Oriented Network Architecture (DONA) [13] from UC Berkeley is considered among the first complete ICN architectures. It replaces naming of data by the hierarchical URLs which are bounded to the location via their DNS component with flat names, and the DNS name resolution with any cast name resolution process. This allows information to be cached and replicated at the network layer, consequently it allows an increasing information availability. In DONA, the authentication and the verification of data integrity is ensured by cryptographic techniques. Besides, DONA maintains the use of IP addressing and routing, either globally or locally, by deploying a name resolution mechanism as an overlay that maps its flat names to the corresponding information.

2.2.1.1 Naming

The names in DONA are flat, application-independent, location-independent and globally unique. The names are organized by using principals, for which is associated a public/private key pair. The name of a piece of data consists of two globally unique principal fields P and L, where P is the cryptographic hash of the principal's public key and L is a label which identifies the object Data. For instance, the principal may name an entire web site or each individual web page within it.

2.2.1.2 Name Resolution and Data Routing

Name resolution in DONA is achieved by specialized servers called Resolution Handlers (RHs). For each Autonomous System (AS) there is at least one logical RH. The RHs are interconnected in a hierarchical manner to form the resolution service. To make the data available, the producer sends a REGISTER message with name of object to its local RH, and in its turn it propagates this registration to the RHs in its parent and neighboring AS. The consumer requests data by sending a FIND message with name of object to the local RH. RH has a registration table to map the incoming request to the destination of the content copy, or to the next hop RH toward the content copy. The REGISTER packet is not forwarded if the RH has already a copy, or if the new REGISTER comes from a copy further away from the previous one. The data can be retrieved either by following the same path of the interest packet in the opposite direction, in this case it is possible to store a copy in the RH caching, or it can be sent back directly toward the consumer.

2.2.1.3 Security

The self-certifying names in DONA allows the verification of the data integrity by the consumer, which removes the necessity for Public Key Infrastructures (PKIs). For mutable data, the hash of the public key and the signature for the data object itself are received with the data object as metadata, while, for immutable data, the label L is the cryptographic hash of the data object. DONA relies on existing IP mechanisms for guarding against user-plane resource exhaustion attacks.

2.2.1.4 Transport

The DONA architecture uses the existing IP transport protocols functionalities to ensure the forwarding and transport functionalities such as flow control, congestion control, and reliability.

2.2.1.5 Caching

In DONA, on-path caching is supported via the RH infrastructure. In order to ensure the coming back of the requested data through current RH to cache it, the RH replaces the source IP address of the FIND message with its own IP address, then the message is forwarded to the next RH.

When a FIND message arrives asking a data object which is found in a caching RH, the RH can directly return the data to the consumer. Data may also be replicated off-path, on condition that each provider registers the information through its local RH. A RH receiving multiple REGISTER messages for the same data object maintains only the pointers to the best available copy.

2.2.1.6 Mobility

When the consumers move, they can simply issue new FIND messages from their new location. The RH infrastructure will provide them with the closest copy of the information. When producers move they can unregister and re-register their information; however, this causes messaging overhead.

2.2.2 PURSUIT (PSIRP)

The PURSUIT architecture consists of three separate functions: rendezvous, topology management and forwarding. When the rendezvous function matches a subscription to a publication, it directs the topology management function to create a route between the producer and the consumer. This route is finally used by the forwarding function to perform the actual transfer of data. The Publish Subscribe Internet Technology (PURSUIT) [14] is an EU Framework 7 Program project launched in 2010 as a continuation to Publish Subscribe Internet Routing Paradigm (PSIRP) [15] (2008-2010), in which a clean slate routing architecture for ICN is proposed. This latter allows to shift the current send-receive based Internet toward the publish-subscribe paradigm. The PURSUIT project essential aim is to develop Internet-scale deployable components of the PSIRP architecture, through producing a new architecture that completely replaces the IP protocol stack with a publish-subscribe protocol stack. The rendezvous function connects a subscription to a publication, it supervises the topology management function to create a route between the producer and the consumer, to be finally used by the forwarding function in order to perform the actual transfer of data.

2.2.2.1 Naming

The names in PURSUIT are flat as in DONA. They are identified by a pair of IDs, the scope ID and the rendezvous ID. The scope ID groups related information objects while the rendezvous ID is the actual identity for a particular piece of information. Scopes control access rights, authorization, reachability, availability, replication, persistence and upstream resources of a content. Information objects may belong to multiple scopes (possibly with different rendezvous IDs), but they must always belong to at least one scope.

2.2.2.2 Name Resolution and Data Routing

The routing infrastructure of PURSUIT consists of: rendezvous, Topology, Routing, and Forwarding components. PURSUIT assumes that the network consists of Autonomous Systems (Domains).

The rendezvous component consists of one rendezvous network per domain. They are interconnected by a global hierarchical (Distributed Hash Table) DHT-based rendezvous interconnect. Rendezvous interconnect matches data sources of certain publications with interests of consumers. The rendezvous network locates the publications and scopes of its network, and advertises its scope to the rendezvous interconnect, so that it becomes globally reachable. The Topology component consists of one topology manager per domain, which manages the intra-domain topology, balances load, and exchanges inter domain path vectors between themselves. The Routing component is built up by a number of branching nodes. These latter use the topology information maintained by a topology manager to route subscription messages from consumers toward data sources and cache popular content.

The Forwarding component has a number of forwarding nodes, which use Bloom filter based forwarding to implement a simple and fast forwarding algorithm to send back a content to the consumer. In PURSUIT, when the producer wants to advertise an information object, it sends a PUBLISH message to its local rendezvous node routed by the Distributed Hash Table to the rendezvous node assigned with the corresponding scope ID. When the consumer sends a SUBSCRIBE message for the same information object to its local rendezvous node, it is routed by the Distributed Hash Table to the same rendezvous node. The rendezvous node then directs a Topology Manager node to create a route that connects the producer to the consumer. This route is sent by the topology manager to the producer in a START PUBLISH message, finally the producer uses this route to send the information object via a set of Forwarding Nodes. In PURSUIT name resolution and data routing are decoupled. The name resolution is performed by the rendezvous network and takes a long time, because the Distributed Hash Table routing does not follow the shortest paths between the communicating nodes, while the data routing is organized by the topology manager and executed by the forwarding nodes and it takes place at line speeds, without placing any state at the forwarding nodes. Furthermore, the separation of routing and forwarding allows the topology manager to calculate paths using complex criteria (e.g., load balancing), without requiring signaling to the (stateless) forwarding nodes. On the other hand, the topology management and forwarding functions as described are only adequate for the intra-domain case and need to be extended for the inter-domain level.

2.2.2.3 Security

In order to encrypt and sign packets, PURSUIT uses the Packet Level Authentication (PLA) technique which ensures data integrity and confidentiality as well as malicious producer accountability. The receiver or forwarding nodes can check the Data packets by using PLA.

As PURSUIT uses flat names, the self-certification is possible for static data, by using hash of object as the rendezvous ID. Moreover, Denial of service (DoS) attacks are impossible to be launched since paths encoded into Bloom filters use dynamic link identifiers, which makes impos-

sible the craft of Bloom filters or even the reuse of the old ones. Finally, for preserving privacy and reducing spam, security solutions have been developed for PURSUIT.

2.2.2.4 Transport

As mentioned previously, the PURSUIT architecture uses basic forwarding process based on the Bloom filters. To handle the flow control, each piece of data has a unique name derived from its original name.

2.2.2.5 Caching

PURSUIT supports both on-path and off-path caching. In the on-path case, in order to serve the future requests of data, the forwarded packets are cached at forwarding nodes. However, since name resolution and data routing are decoupled in PURSUIT, on-path caching may not be very effective, because the requests of the same data can reach the same rendezvous node, whereas the actual data deliveries may use entirely different paths. In the off-path case, caches operate as producers, by advertising the available data to the rendezvous network. In the PURSUIT architecture, managed information replication, can also be efficiently supported as in CDNs.

2.2.2.6 Mobility

In PURSUIT, the mobility is enormously facilitated by using multicast and caching. Local consumer mobility can be handled via multi-cast and caching, since the mobile consumer can ask and receive multi-casted data in multiple locations from nearby caches after a hand-off. Producer mobility is harder, because the producer's new position in the network must be notified to the topology management function.

2.2.3 NetInf

Network Information (NetInf) allows adaptation to different network environment by offering two models for retrieving NDOs (Named Data Object): via name resolution and via name-based routing. Depending on the used model in the local network, content is published by registering a name/locator binding with a Name Resolution Service (NRS), or use a routing protocol to announce the routing information. Via name resolution the consumer sends the request to the NRS, this returns the available locaters of the named data object, thus the consumer retrieve the data from the best available sources. Otherwise, via name-based routing the client sends directly the request, which is forwarded to the source, besides the data is sent to the consumer when a NDO is reached. The two models can be used separately or merged in a hybrid scheme where it is possible to switch between the two schemes based on hop by hop feature.

2.2.3.1 Naming

In NetInf, flat namespaces are used for naming in a structure which is similar to DONA architecture [16]. NetInf naming aims to make difference between a common naming format that all nodes can understand it, ensure information security to maintain integrity of the data, and differentiate the name-object binding validation mechanisms. The name format of the NetInf supports different hashing schemes. The common NetInf naming format [17] is based on containing hash digests in the name, and different hashing schemes (e.g., SHA2-based object content digests) are supported. The hash digest of the owner's public key (PK) can also be contained in the name to support dynamic data. The different naming representation are supported In NetInf, such as Uniform Resource Identifier, and binary representation.

2.2.3.2 Security

As DONA names in NetInf are self-certifying, so static and dynamic contents are secured. The naming format of NetInf allows the data-integrity validation by nodes. And also the named data is validated without a Public Key Infrastructure. Furthermore, content security is ensured via the public key cryptography.

2.2.3.3 Transport

Different forwarding mechanisms are used in NetInf for retrieving a data object, locator or redirection hints by using a set of messages to request/response resources. This protocol is implemented by convergence layers, which provide a concrete abstraction for specific underlays. The communication is ensured by a hop-by-hop approach in which the convergence layer implements a special transport protocol that provides the appropriate resource sharing and reliability mechanisms for the corresponding network path.

2.2.3.4 Caching

NetInf can cache requests and objects. When a node receives a GET request, it can decide to employ whether a pending interest-table-like structure for request aggregation or perform a dedicated NRS lookup for each received interest, thus performing request aggregation becomes a policy decision. Two ways for using a stored content copy: first, any copy can be retrieved directly from the NRS, provided that the copy is registered there or discovered via other means; second, the copy can be found by a cache-aware NetInf transport protocol on the path to a location known to hold a copy [18].

2.2.3.5 Mobility

In NetInf consumer mobility is ensured by its indirection between identifiers and locators. The exact details of this vary based on the chosen locator selector mode [19]. In the requester-controlled mode, a list of potential sources is provided to the consumer, thus allowing a node to select a new optimal source following re-location. While in the MDHT-controlled mode, the consumer receives only a single source for each request, and mandates a re-located node to contact the NRS again. In both modes mobility should be enabled, by assuming fast lookups. Concerning the producer mobility in NetInf, it is more difficult because it requires the NRS to be updated; however, it is claimed that updates can be scalable handled [20].

2.2.4 MobilityFirst

The MobilityFirst [21] project, is one among five projects funded by the US Future Internet Architecture program. It proposes a new architecture for the next generation Internet according ICN paradigm. Its main feature is that it is based on affecting high importance to the mobile devices. Consequently, MobilityFirst proposes detailed mechanisms for handling both mobility and wireless links, as well as multicast, multi-homing, in network caching and security. MobilityFirst principle is to decouple the names for all entities attached to the network from their network addresses such as information objects, devices and services. So, in order to allow messages to be dynamically redirected and follow a mobile device or content, each entity possesses a globally unique name, which can be transferred into one or more addresses at various points.

2.2.4.1 Naming

In MobilityFirst a Globally Unique Identifier (GUID) is assigned to each entity in the network ensured by a global naming service. This latter is responsible to translate human-readable names to GUIDs. Every device, its information objects, and its services in MobilityFirst must have GUIDs, which are flat (160-bit strings). Since GUIDs length ensures that the probability of a collision is small, they do not have semantic structure and they may be randomly selected. MobilityFirst can support both name-based information delivery (via information GUIDs) and host-to-host communication (via device GUIDs) by naming all network entities.

2.2.4.2 Name Resolution and Data Routing

MobilityFirst uses a hybrid process between the IP routing and name-based routing for name resolution and data routing. By using the Global Name Resolution Service (GNRS) to map the GUIDs to IP addresses, the routing in MobilityFirst is performed by IP address.

In case of less dynamic services, each GUID can be translated one time to a IP address, as in DNS, and then the routing is based only on this address, so the GUID is ignored.

In case of more dynamic services, each GUID can be translated to IP address multiple times, at any router in the path. The GNRS is asked for the IP addresses that are bounded to a given GUID and forwarding decisions are made based on the reply received from GNRS.

So, when the GNRS is bypassed the forwarding is ”fast path”, and it is ”slow path”, when the GNRS are repeatedly consulted by routers in order to obtain an updated list of IP addresses. This latter known as late binding is especially useful to handle the issue of host mobility.

In MobilityFirst the request packets (GET message) and the corresponding Data packet are routed based on their destination GUIDs, i.e. name resolution and data routing are decoupled in this architecture.

When the producer wants to make some information available for retrieving, it asks the naming service for a GUID and then registers it with its network address in the Global Name Resolution Service. So, The Global Name Resolution Service maps the GUID via hashing to a set of IP server addresses.

When a consumer desire some data, it sends a GET message with the GUID of the requested data, to its local Content Router. The Content Router asks the GNRS for a mapping between the destination GUID and its corresponding IP addresses. The Content Router, adds one of the received addresses to the GET message, which is forwarded according routing tables in the Content Routers.

In MobilityFirst the GET message carries both the destination GUID and the destination network address. Any Content Router along the path can consult the GNRS to receive an updated list of network addresses for the destination GUID in the case of slow path.

2.2.4.3 Security

In MobilityFirst, the GUIDs can be self-certifying hashes of information objects, which allow the verification of information integrity, or hashes of public keys, that allow binding devices to principals. On the other hand, to avert profiling users can frequently request a new GUID. MobilityFirst uses independent naming organizations for mapping human-readable names to GUIDs in order to envision a decentralized trust model for name certification.

2.2.4.4 Caching

MobilityFirst supports on-path caching and off-path caching. The intermediate Content Routers can store the received messages in order to satisfy the future requests for the same GUID. In addition, each time a Data packet is replicated, the Global Name Resolution Service updates the corresponding GUID entry with the additional network addresses after being informed of the change. When a message travels through the network, the Global Name Resolution Service can be repeatedly consulted in a “slow-path” operation. In addition, each Content Router can implement its own policy on when consult with the Global Name Resolution Service for additional cached copies.

2.2.4.5 Mobility

In MobilityFirst, disconnections due to mobility and variable link conditions are handled by Global Name Resolution Service which must be updated when a network attached object changes its point of attachment, so when the producer or the consumer moves, its area is firstly localized and then its location is defined, via the late binding and transmission of packets can be ensured.

2.2.5 Named Data Networking

The proposed ICN architectures in literature use different approaches but all of them focus on the same principle, which is based on content delivery rather than on host-centric. These architectures have known more or less success, depending on the complexity of their implementation. On a practical point of view, few of these architectures are declared to be applicable. Named Data Networking (NDN) [8] is one of the most popular ICN proposals, and it is one among the five projects founded by the US National Science Foundation under its Future Internet Architecture Program. NDN is the successor of Content-Centric Network (CCN) that has been presented publicly for the first time by Van Jacobson in Google Talk in 2006 [12]. The first paper that described the CCN architecture was published in 2009 [7]. The NDN inherits the hourglass shape of the IP architecture, but replaces the end-to-end data delivery model at the thin waist by a receiver-driven data retrieval model as seen in Figure 2.1. This fundamental change leads to the shift of communication paradigm from location-centric (Where) to data-centric (What).

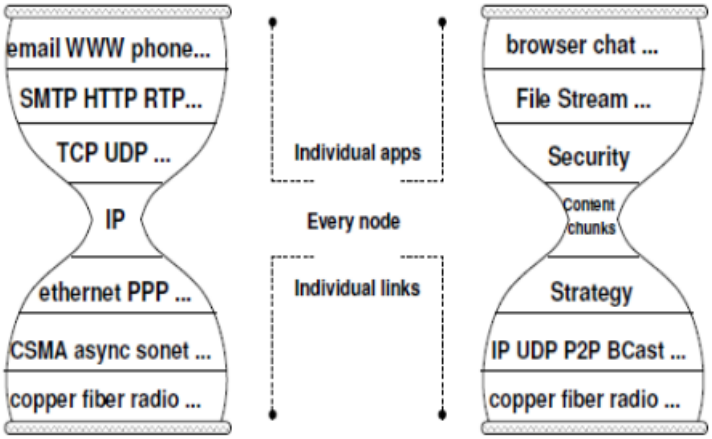


Figure 2.1: NDN and IP stack, from [7].

Given the advantages of NDN architecture such as the simplicity, efficiency, accessibility (Open-source implementation CCN and NFD), and large community. Furthermore, about half of the whole ICN software ecosystem are dedicated for CCN/NDN [**?**]. Consequently, the contributions of this dissertation are designed according NDN architecture.

2.2.5.1 NDN node architecture

Unlike IP networks, where the semantics of network services are delivering the packet to a given destination address, the NDN services semantics are retrieving data identified by a given name. The name in an NDN packet can refer to: an endpoint, a data chunk in a movie or a book, etc. All NDN communications are consumer-driven based on two types of packets: Interest and Data. The former is a small packet similar to TCP acknowledgments, generated by a consumer to request the desired data: the consumer puts the desired data name into an Interest packet and sends it in the network. Each Interest packet also carries other very important fields for the search of the requested data such as a selector field which provides more specific descriptions of the desired data, and a nonce field which is a random number generated by the consumer which used to detect and avoid looping packets. A Data packet is transmitted only in response to an Interest and consumes that Interest. The Data packet can be generated by any node having the requested data, like the producer or in-network storage: persistent storage (repository) or temporary storage (caching router). In fact, in NDN, a router can cache received Data packets in its content store and use them to satisfy future requests. The Data packet is used to carrier the requested data with some additional information such the cryptographic signature that binds the data to the name which used to protect the content. Figure 2.2 shows the data structure of Interest and Data packets.

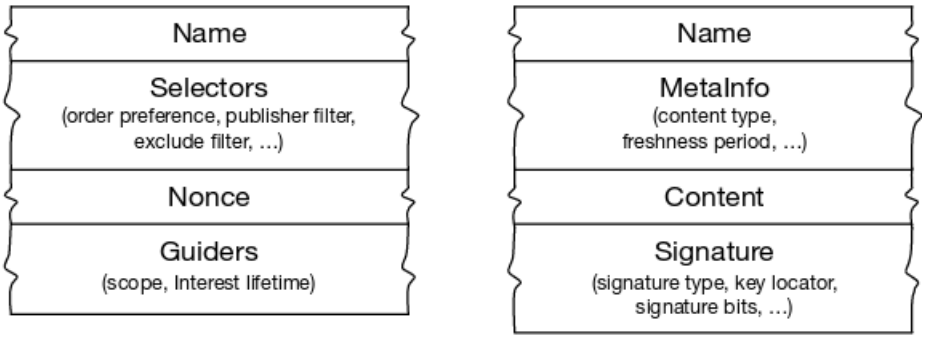


Figure 2.2: NDN architecture packets, from [9].

Each NDN node has three data structures: CS (Content store), PIT (Pending Interest Table) and FIB (Forwarding Information Base).

Content Store (CS): The Content store is used to store temporarily the Data packets received at the node, which allows the node to speed up the satisfaction of future Interests. The role and advantage of CS are described in Section 2.2.5.6.

Pending Interest Table (PIT): In NDN, an Interest packet carries neither address nor name to identify the requesting consumer that can be used to return the requested Data packet. Instead

NDN routers use Pending Interest Table (PIT) to keep track of incoming interfaces for pending Interests, which are forwarded upstream toward potential data source(s), but not yet satisfied. These information are used to bring matched Data packets back to consumers. The PIT table is indexed by name. So, for each requested name a PIT entry is created. A PIT entry contains a list of nonces that have been seen for that name, a list of incoming interfaces from which Interests for that name have been received, as well as a list of outgoing interfaces to which the Interest has been forwarded. In a PIT entry, each incoming interface records the longest Interest lifetime it has received; when the lifetime expires, the incoming interface is removed from the PIT entry, and the entire PIT entry is removed when all its incoming interfaces have been removed. Each outgoing interface records the time when the Interest is forwarded via this interface, so that when Data packet returns, Round Trip Time (RTT) can be computed. The RTT measurement is then used to update the RTT estimation for the corresponding name prefix stored in the FIB [22] [23]. Figure 2.3 shows the PIT entry data-structure that maintains datagram forwarding state [22] [23].

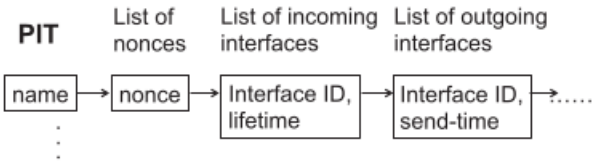


Figure 2.3: Forwarding State in PIT, from [24].

Forwarding Information Base (FIB): A FIB of NDN router is similar to the FIB in an IP router except that it contains data name prefixes instead of IP address prefixes. So, the FIB stores the data name prefix used to forward Interest packets toward requested data potential source(s). An IP FIB entry usually contains a single best next-hop and its routing information, while an NDN FIB entry contains a ranked list of multiple interfaces and record for each interface information from routing and forwarding planes, providing input to forwarding decisions. In addition, to a three data structures CS, PIT and FIB, each NDN router is also equipped with a Forwarding Strategy module, that makes the forwarding decisions for each Interest packet according to the information stored in these data structures that used as input for the algorithm adopted by forwarding strategy. For more details see Chapter 3.

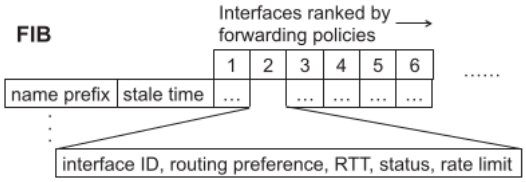


Figure 2.4: Forwarding State in FIB, from [24].

2.2.5.2 Forwarding Process

In NDN network, the nodes process the Interest and Data packets as described in Figure 2.5.

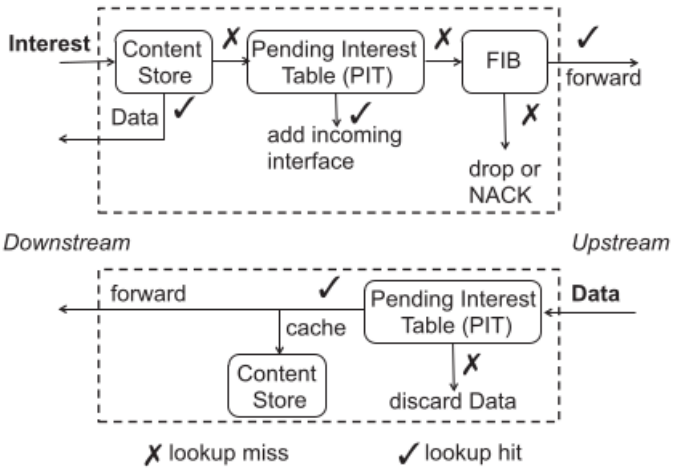


Figure 2.5: Forwarding State in FIB, from [24].

When a router receives an Interest packet, it first checks the Content Store (CS) for matching data. If matched data is found, then it will be sent back via the incoming interface of the Interest and the Interest will be dropped as it is considered satisfied. Otherwise, the router looks up in PIT entries. If an entry is found, it can be either a duplicate Interest that should be dropped, an Interest retransmitted by the consumer that may need to be forwarded using a different outgoing interface, or an Interest from another consumer asking for the same data which requires the incoming interface of this Interest to be added to the requesting interfaces list in the existing PIT entry, then it drops the Interest. When the requested data is found, a copy will be sent back via incoming interfaces of Interest.

If the Interest name does not exist in the PIT, a router creates a new PIT entry for the incoming Interest and looks up for the Interest name in the FIB table. If no FIB entry matches the Interest name, the Interest is deleted. Otherwise, the incoming Interest will be forwarded via the interface selected by the forwarding strategy module.

When a router receives a Data packet, it looks up in the PIT entries. If a matching PIT entry is found, the router sends the Data packet to the interface(s) from which the Interest was received and removes the PIT entry. Consequently, the Data packets always take the same paths of Interests but in the reverse direction. If no match is found, the Data packet is unsolicited and discarded. Each Interest also has an associated lifetime set by the consumer; a PIT entry is removed if the Interest is not satisfied before its lifetime expires.

A router can store the received Data packet in the CS according to its caching policy. In-network caching is useful, since it gives a possibility of retrieving the requested data directly from router

without going to the producer, which allows to reduce the data delivery time. Furthermore, it is possible to satisfy the consumer retransmitted Interest, when congestion losses occur, from the requested data that has been cached in one router of the path, before it gets dropped. Thus, we can avoid the transmission of the Interest up to the producer, since it is probably that the path towards the producer may be still congested.

2.2.5.3 Naming

In NDN, the routers guide the Interests packets toward the potentials data source by exploiting data names. The names in NDN are used to identify a unique piece of data which can be carried by one packet [6]. Data satisfies an Interest if the data name in the Interest packet is a prefix of the data name in the Data packet [7].

The names in NDN are hierarchically structured, composed of multiple components. A component can be any string of arbitrary length, the delimiter '/' delineates name components in text representations, similar to URLs and shows the hierarchy of the name component. For example Figure 2.6 shows the application-level conventions currently used to capture temporal evolution of the content (a version marker, $\_v$ encoded as FD, followed by an integer version number) and its segmentation (a segment marker, $\_s$ encoded as 00 followed by an integer values which might be a block or byte number or the frame number of the first video frame in the packet) [7]. Furthermore, NDN names are opaque to the network. In other words, the routers do not understand the meaning of a name but recognize boundaries between components in a name. The hierarchical structure of names has three advantages. First, the applications can define their own naming schemes that fits their needs and allow naming schemes to evolve independently from the network. Second, the applications can put the context and relationship of the data elements into the names. Finally, it allows name aggregation which is essential in scaling the routing system. In the example shown in Figure 2.6 *parc.com* could be the autonomous system from which the video originates.

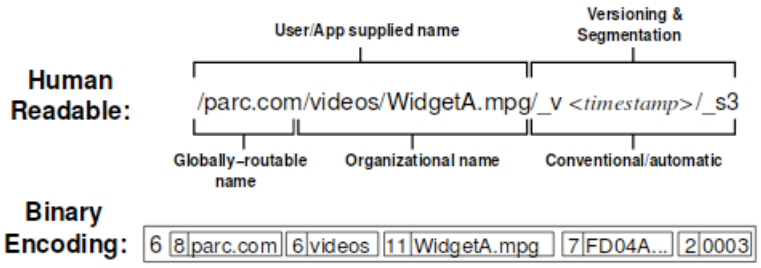


Figure 2.6: Example Data Name, from [6].

To retrieve dynamically generated data, the name for the desired piece of data must be able to be deterministically constructed by the consumers, without having previously seen the name or the data. Either 1) a deterministic algorithm allows the producer and consumer to arrive at the same

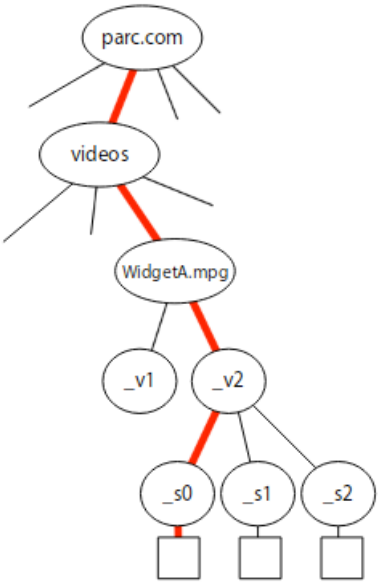


Figure 2.7: NDN node data structure, from [6].

name based on information available to both, or 2) Interest selectors in conjunction with longest prefix matching retrieve the desired data through one or more iterations [25].

In [25] it has been suggested that a simple set of selectors can support retrieving data with partially known names. For example Figure 2.7 shows a portion of the name tree associated with Figure 2.6, a consumer wanting the recent version of the *WidgetA.mpg* video may request */parc.com/videos/WidgetA.mpg* with the Interest selector "Rightmost child" and receive a Data packet named */parc.com/videos/WidgetA.mpg/*$_{V_2}$*/*$_{S_0}$ corresponding to the second first segment of the second version of the video. Then, the next segment is obtained by sending an interest containing its name with a LeftmostRightSibling annotation or by computing the $_{S_1}$ portion of the name since the segmentation rules are known by the consumer [25].

2.2.5.4 Security

In IP architecture, the security (or lack thereof) depends on where and how the data is obtained. While the NDN secures the data themselves, i.e. the content is secured instead of channel and connection, so the producers secure contents by signing cryptographically with its secret key every piece of data, which ensures data integrity. As result the consumers can determine easily the data provenance, allowing their trust of data to be independent from where and how the data is obtained. Since the names in NDN are human-readable hierarchical structured and do not contain the producer's public key and self-certification is not possible. Furthermore, security based on public key

cryptography is typically considered inefficient as well as difficult to deploy and use.

In NDN architecture, the data-centric security has natural applications to content access control and infrastructure security. Applications can control access to data via encryption and distribute (data encryption) keys as encrypted NDN data, limiting the data security perimeter to the context of a single application. Requiring signatures on network routing and control messages (like any other NDN data) provides a solid foundation for securing routing protocol against, e.g., spoofing and tampering [25].

Since NDN supports inherently multipath routing, with its powerful forwarding plane, it can reduce prefix hijacking due to the routers ability to detect the anomaly caused by a hijack and retrieve data via alternate paths.

2.2.5.5 Transport

The NDN architecture does not have a separate transport layer. The transport protocols functionalities of Today are provided by applications, their supporting libraries, and the forwarding strategy. Multiplexing and demultiplexing among application processes is done directly using names at the NDN layer, and data integrity and reliability are directly handled by application processes where the appropriate reliability checking, data signing and trust decisions can be made [8]. Furthermore, the hierarchical namespace allows the required information for transport to be included in the data names, consequently eliminating the need for transport layer information such as sequence and port numbers.

A NDN network is designed to operate on top of unreliable packet delivery services, including the highly dynamic connectivity of mobile and ubiquitous computing [8]. To provide reliable, resilient delivery, the application itself or its supporting library will monitor the Interest packets that are not satisfied within some reasonable period of time and retransmit them if it still wants the data.

In NDN architecture, the flow balance requirement, together with the ability of nodes to control their own traffic load by limiting the number of pending Interests at each hop can provide effective congestion control throughout the network [9].

To limit the congestion in the network, each Interest packet has a limited lifetime. If congestion losses occur, caching can reduce the impact since retransmitted Interests can be satisfied by cached Data packets in routers that precede the point of packet losses, while in IP architecture, the retransmission of data will happen all the way back at the destination, and the packet has to try to pass the all the path again. Thus, NDN avoids congestion collapse that can occur in today's Internet when a packet is lost near its destination and repeated retransmissions from the original source host(s) consume most of the bandwidth. In NDN, data make steady progress towards the final destination, as the cached copy is used to satisfy the original Interest as well as retransmitted Interests.

2.2.5.6 Caching

Caching is one of the most important features of the ICN architectures [12]. In NDN networks, the Data packets are independent from the requesters and from where it stored, since each Data packet carries the data name and signature instead than IP source and destination in IP architecture. So, a router can store the received Data packets in its Content Store (CS) and use them to satisfy the future requests. Upon receiving a new Interest, the router first checks the CS for matching data; if it exists the router returns the data on the interface from which the Interest came.

Caching the content in CS is analogous to buffer memory in IP routers, the difference is that IP routers cannot reuse the data after forwarding them, while NDN routers are able to reuse the data since they are stored in routers CS and identified by persistent names. NDN achieves almost optimal data delivery for static files. Even dynamic content can benefit from caching in the case of multicast, such as real time teleconferencing or retransmission after a packet loss [7].

The CS cannot store a Data packets for long time, so it uses different replacement policies such LRU, FIFO, etc. The caching is mostly useful for recovery from packet losses and handling flash crowds, where many users request the same data in close succession. In addition to the Content Store, the NDN architecture supports more persistent storage with a very large and larger-volume in-network storage, called Repository; which may offer services similar to those of the Content Delivery Networks (CDN). Although NDN can support more powerful CDN architectures than TCP/IP, NDN also provides many other functions (securing data, flow balance, state full data plane which leads to a number of gains on its own) that present even more significant and important advantages [9].

Caching named data raises different privacy concerns from those of IP. In IP architecture, by examining packet headers or possibly payload, it is possible to identify the requester and the requested data and its emplacement. However, in NDN, as the data name does not have any information of requester and source of requested data, it is harder to identify the source of data and requester. So, NDN architecture offers some natural privacy protection by making data and users more secure.

2.2.5.7 Mobility

The NDN properties facilitate mobility since data names are fixed and do not change with mobility, the data are granular and sessionless. So, when a consumer moves, the corresponding data objects may go to the old location. The consumer can simply send a new Interest for the data objects that it has not yet received from its current location upon timeout. It is possible to get the requested data from an intermediate cache. When a producer moves, it is possible to satisfy the Interests for data in moved producer by intermediate caches. In addition, the moved producer must announce the name prefixes for the contents that it hosted via the routing protocol at the new location subsequent Interests will go to new location. As this represents a very high overhead in high-mobility solu-

tions, NDN utilizes the Listen First Broadcast Later (LFBL) protocol [26] to implement mobility in ad-hoc/opportunistic networks. In LFBL, Interests are flooded. When a potential source for the requested data receives an Interest, it listens to the (wireless) channel in order to discover if another node has already sent a matching data. If not, it sends the data itself towards the consumer.

2.3 Comparison between the various approaches

The aforementioned ICN architectures have similarities and differences in the techniques used for implementing the key ICN functionalities. Tables 2.1 summarize the techniques used by each approach for handling the naming, name resolution and routing, caching, mobility, name data object granularity and security.

	Naming	Name Resolution and Data Routing	Caching	Mobility	Security	Name Data Object granularity
DONA [13]	Flat, consisting of principal and label part. Not human-readable	Resolution handlers organized following AS hierarchy. Coupled: A source-route is created during resolution. Decoupled: Resolution handlers return network address.	On-path caching at resolution handlers. Off-path caching requires additional registrations.	When consumer mobility, it makes new requests. When producer mobility, it requires additional registrations.	Principal is hash of public key. Label can be hash of Static content.	objects
PURSUIT [14]	Flat, consisting of scope and rendezvous part. Scopes may be organized Hierarchically. Not human-readable	DHT-based rendezvous network matches subscriptions to publications. Scopes can be used to limit publication/resolution. Decoupled: topology management and forwarding are separated from rendezvous.	On-path caching is difficult due to decoupled operation. Off-path caching requires additional registrations.	When consumer moves, it makes a new requests. when producer moves, it requires updating the topology manager.	Packet level authentication for individual packets. Names can be optionally self-certifying.	objects
Mobility First [21]	Flat, consisting of a single Component. Not human-readable	Hash-based global name resolution service to map names to IP addresses, may be used repeatedly for late binding of addresses. Decoupled: requests and data are independently resolved and routed.	On-path caching at content routers. Off-path caching requires additional registrations.	When consumer or Producer move, they reach the area mobiles via late binding to first and then find the mobile.	Names can be optionally self-certifying.	objects
NetInf [18]	Flat, consisting of principal and label part. Not human-readable	Name Resolution Service used to publish Information Objects alongside their locator(s) to the name resolution service for later discovery by consumers;	On-path caching at location known to hold a copy. Off-path caching any copy can be retrieved directly from the NRS, provided that the copy is registered there or discovered via other means	As consumer is provided with a list of potential sources, that allows to select a new optimal following re-location. For producer mobility, it is more difficult because it requires the name resolution service to be updated.	Signature or content hash, PKI independent	objects
NDN [8]	Hierarchical, may contain producer specific prefix. Can be human-readable	Routing protocol used to announce name prefix Information. Coupled: Routing state for data is established at routers during request propagation.	On-path caching at content routers. Off-path caching requires additional Routing information.	When consumer moves, it makes new requests. When producer moves, the Interest are flooding.	Signatures included in packets. Certification chain can follow name hierarchy.	Packets

Table 2.1: Summary of the ICN approaches.

2.4 Conclusion

In this chapter, we have described five ICN architectures proposed in different projects: DONA, PURSUIT, NetInf, Mobility First, and NDN. However, there exist other architectures that are not described such as: CONVERGENCE [27], Sail [28], COMET [29], 4WARD [30], and ANR CONNECT project funded by The French National Research Agency Projects for science [31] which adopts NDN architecture. Although, the aforementioned architectures propose different approaches, they share the same design paradigm which is centric-content rather than centric-host. In practicability side of view, few of these architectures are declared to be applicable.

We have detailed more the NDN architecture as our contribution are designed according to it. In the next chapter we will investigate the NDN routing issue and we will present the literature related to this topic.

Chapter 3

Related Work

Contents

3.1 Introduction 49
3.2 Traditional Routing in IP 50
3.3 Routing Plane in NDN 51
3.4 Forwarding Plane in NDN 51
3.5 Routing and Forwarding Coordination in NDN 52
3.6 Building the FIB table 53
3.7 NDN Routing Protocols: a survey 55
3.7.1 OSPFN: OSPF for Named Data Networking 55
3.7.2 NLSR: NDN Link State Routing 56
3.7.3 Hyperbolic routing 59
3.8 QoS based routing in NDN network 61
3.8.1 Problem Formulation 61
3.9 Forwarding Strategies: a Survey 62
3.9.1 High level taxonomy 62
3.10 Conclusion 67

3.1 Introduction

In Chapter 2, we have presented the ICN paradigm for designing future Internet architecture, its principles and objectives. Furthermore, we have introduced an overview about the ICN fundamental architectures proposed by the various ICN projects. We have focused more on NDN architecture since our contributions are designed according to it. In this chapter, we investigate the routing and forwarding issues in the NDN architecture. The packet delivery system in NDN is organized in

two complementary planes like in IP networks: the routing plane and the forwarding plane. In this section, we will explain the role of each plane and the difference between them and we present the related literature for routing, forwarding strategies and congestion challenges.

3.2 Traditional Routing in IP

In IP architecture, the routers route and forward the packets according to the IP source and destination address. The routing plane in IP networks serves to announce IP prefixes, an IP router announces IP prefixes of networks connected to it. Each router announces the IP address of hosts and networks connected to it, and computes the paths. The routing protocols such as Link State or vector distance, OSPF, BGP, etc. are responsible for propagating of announcements through the network, and every router builds its FIB based on received routing announcements and compute the best next hop. Thus, the IP RIB (Routing Information Base), which corresponds roughly to the FIB table in NDN entry usually contains a single best next hop and its information. More specifically, in IP networks the routing plane is responsible for populating the RIB tables of routers and keeping them updated in face of network changes, including both long-term topology and policy changes as well as short-term churns. As result, the routing plane is the control plane, it is state-full, intelligent and adaptive manner. It performs all the work, when the forwarding plane is dumb, one-way traffic, and stateless. Furthermore, at the data plane, routers forward packets strictly following the RIB table via single interface [22]. Consequently, the robust packet delivery in IP networks depends only on the routing plane.

While the routing plane is responsible of all the work in IP networks, when there is a change in the network i.e. failure detection, routers need to exchange routing updates between them in order to achieve new global consistency. The time period between the instant when a failure is detected at the network and the instant when all routers are updated to a new situation is called the routing convergence time. The routing convergence time can be divided into four different components: failure detection, update propagation, route computation and RIB update. As result, in order to reduce packet loss and resume packet delivery after network is updated, IP routing protocols need to converge fast. However, fast routing convergence is challenging in large operational networks, because it conflicts with routing stability and scalability i.e. it is very difficult to reach a fast convergence, stability and scalability simultaneously.

The routing stability is very important for applications that suffer from RTT fluctuation. Scalability is necessary to support a large network composed from a big number of routers, links and IP prefixes. For example, the distance/path-vector routing protocols, where routers keep only the hops count between them as information that used for route and forward packets, without full knowledge of the topology ensures a better scalability, but the convergence time may be long (tens of minutes). Link-state routing protocols, however, as each router knows the entire topology, can converge fast,

but it suffers from poor stability and limited scalability.

Concerning the multi-path routing, in IP network it is considered as challenging, because the IP architecture does not support multi-path routing inherently. Furthermore, multi-path routing needs to maintain routing consistency to make sure there will be no loops. Consequently, the routing plane is responsible to guarantee loop-free paths by the fast convergence when it detects looping packets.

3.3 Routing Plane in NDN

In NDN network, the routers route and forward the packets according to names, which eliminates the problems related to IP addresses. First, there is no address exhaustion problem since the namespace in NDN is unbounded. Second, there is no NAT traversal problem since the routing system does not requires for the hosts to expose its address private or public in order to offer content, but the routers guide back the retrieved data to its requesters according to the interfaces tracks stored in its PIT data structure. Finally, address assignment and management is no longer required in local area networks [9].

The routing plane in NDN serves the same goal like in IP networks. Instead of announcing IP prefixes, a NDN router announces name prefixes that cover the data that the router is willing to serve. The content providers announce the names prefixes and the other routers compute the paths. This announcement is propagated through the network via a routing protocol, and every router builds its FIB based on the received routing announcements [8]. Indeed, the routing plane is responsible for populating the FIB tables of routers and keeping them updated. Any routing scheme that works well for IP should also work for NDN, because forwarding model of NDN is a strict superset of the IP model [6]. So, the conventional IP routing protocols such as link-state, OSPF, etc. can be used in NDN networks. It is worth mentioning that until now there is none standardized NDN routing protocol.

Since NDN network has an intelligent and adaptive data plane, the design of the NDN routing plane can be different from today existing routing plane, as it only helps the forwarding plan by populating the FIB.

3.4 Forwarding Plane in NDN

The data plane in NDN network is composed of two-step process: consumers send Interest packets, then Data packets flow back across the same path but in the opposite direction. Therefore, *adaptive forwarding* is enabled by this symmetric Interest-data packets exchange and forwarding state maintained at routers [23] [22]. Since, by recording pending Interests and observing Data packets coming back, each NDN router can measure packet delivery performance such as round-trip time and throughput, and can detect the problems that lead to packet losses such as link failures or con-

gestion. For example, the requested data is supposed to come back within reasonable time, so if a router does not receive the requested data from the same interface from where the Interest was forwarded in this period (time out), that means something is not right: dropped due to congestion, link failures, loops, etc. So, the forwarding plane can bypass problematic areas by using multiple alternative paths. It is worth mentioning that in [23] [22] it is introduced for the first time the interest **Nack**. So, when an NDN node can neither satisfy nor forward the incoming Interest, it generates and sends back to the downstream node an Interest **Nack**, to notify the network problem quickly. An Interest **Nack** carries the same name as the incoming Interest, plus a code that explain why it was generated.

The forwarding plane can detect and recover from network failures on its own. Indeed, it enables each router to handle networks failures locally and immediately without waiting for the global routing convergence [32]. The forwarding plane can also detect and handle looping packets issue, since each Interest carries a nonce field (a random number) which, together with the Interest name, uniquely identifies the Interest. Furthermore, each router records the nonce of received Interest in the matching PIT entry; hence if the incoming Interest, name plus nonce together, has a matching record in PIT, the incoming Interest is looped and a router drop it. In addition, since the Data packets follow the same paths crossed by the corresponding Interests but in the reverse direction, the Interests packets cannot loop either. As result, an NDN router can forward an Interest using multiple interfaces without worrying about loops.

One of the most important features of NDN is the inherent support of multipath routing ensured by forwarding plane, which prevents the network from packets looping. NDN routers can forward a received packet toward multiple paths i.e. interfaces. Consequently, NDN FIB entry contains a ranked list of multiple interfaces and it records its routing and forwarding planes information. The received interest is send out toward the interface(s) selected by the forwarding strategy module based on routing information, forwarding information, and forwarding policies set by the operator that must take into consideration all NDN features, especially caching.

3.5 Routing and Forwarding Coordination in NDN

As we explained in previously section, NDN inherently supports multipath routing. Consequently, each NDN FIB entry contains a ranked list of multiple interfaces and records its routing and forwarding planes information. The former determines which paths are available to forwarding plane, while the latter determines which paths may be used and in which order. At first time the interfaces are ranked according to routing information, but subsequently the interfaces will be ranked according to both routing and forwarding information. These information are considered as input for the forwarding strategy module, which used them and the coordination between them in incoming Interest forwarding process through the optimal paths toward the request data.

The coordination between routing and forwarding is discussed in [32]. The authors raise the question of whether routing protocols are still needed in NDN. If so, what impact may an intelligent and powerful forwarding plane have on the design and operation of NDN routing protocols and to what extent? They found, through extensive simulations and analysis that a routing protocol is highly beneficial in bootstrapping the forwarding plane for effective data retrieval, efficient probing of new links or recovered links, because routing protocol provides information that reduce probing upon searching the working path. Furthermore, since the forwarding plane is able to detect and recover quickly from network failure on its own, NDN routing does not need to converge fast according to network changes. In addition, routing stability and scalability can be significantly improved, because NDN routers can handle network failures at the forwarding plane, and the routing protocol can be freed from short-lived failures. Based on the coordination between NDN routing and forwarding plane, the routing algorithms that would not work fine in IP network may work fine in an NDN network.

3.6 Building the FIB table

In NDN, in order to make the data stored on producer available for retrieval. The producer application firstly registers the prefix of data with the NDN forwarder on the same host machine (i.e., local registration). Secondly, it signals available data, in order to be fetched by remote NDN forwarders. There are many ways to achieve this task: by using the routing protocols such as NLSR, OSPFN, etc. as explained in Section 3.3, or manual configuration, or opportunistic data discovery strategies (e.g., the access strategy, and auto-registration of the specified prefix(es) upon creation of on-demand faces). These methods have different tradeoffs in ease of use, implementation complexity, and communication overhead. To address the aforementioned issue an Automatic Prefix Propagation (APP) protocol is proposed in [33]. Automatic Prefix Propagation (APP) enables the local NDN forwarder in one producer to automatically propagate local prefix registrations to the remote forwarder on attached router. Local prefix registration triggers the propagation when an active remote NDN gateway exists, while the local forwarder possesses a private key and the corresponding NDN certificates that it matches or covers the locally registered namespace. In the case of two or more candidates for the keys/namespace, the selected key corresponds to the shortest namespace. This allows the forwarder to aggregate multiple local registrations into one remote action, reducing communication and bookkeeping overheads. After the initial prefix propagation, NFD (NDN Forwarding Deamon) periodically refreshes the registration, unless the prefix is no longer registered locally or there are no NDN gateways configured.

To implement and validate our proposals, the ndnSIM simulator for ns3 [34] has been chosen; because the studies performed in [**?**] about ICN software tools show that the ndnSIM represents a fairly good compromise between completeness and scalability. Also it is the best one for security

and routing issues among CCN/NDN simulators.

In ndnSIM simulator for ns3 [34], the FIB table is built as follow: all forwarding strategies use the FIB (Forwarding Information Base) table for forwarding Interests. By default, all nodes have empty FIB. They need either to manually configure routes, use global routing controller, or (not recommended) enable default routes.

- **Manually routes (FIB Helper)**
  The FIB helper interacts with the FIB manager of NFD by sending special Interest commands to the manager in order to add/remove a next hop from FIB entries or add routes to the FIB manually (manual configuration of FIB).

  Adding a route to the FIB manually:
  Ptr<Node> node = ... // some node
  std::string prefix = ... // some prefix
  Ptr<ndn::Face> face = ... // NDN face that belongs to the node and through which prefix is accessible
  $int32\_t\ metric$ = ... // some routing metric
  FibHelper::AddRoute(node, prefix, face, metric);
  FIB helper has few other *AddRoute* overloads that may be easier to use. For example, when setting up manual routes between nodes connected with PointToPointNetDevice, it is simpler to use the overload that accepts two nodes (face will be automatically determined by the helper).

- **Automatic Shortest Path Routes (*Global Routing Helper)***
  To simplify FIB management in large topologies, ndnSIM contains a global routing controller (helper and special interface), similar in spirit to *Ipv4GlobalRoutingHelper*. There are several necessary steps, in order to take advantage of the global routing controller: install special interfaces on nodes
  NodeContainer nodes;
  ...
  GlobalRoutingHelper ndnGlobalRoutingHelper;
  ndnGlobalRoutingHelper.Install(nodes);
  specify which node exports which prefix using *GlobalRoutingHelper::AddOrigins()*
  Ptr<Node> producer; // producer node that exports prefix
  std::string prefix; // exported prefix
  ...
  ndnGlobalRoutingHelper.AddOrigins(prefix, producer);
  calculate and install FIBs on every node using *GlobalRoutingHelper::CalculateRoutes()*
  GlobalRoutingHelper::CalculateRoutes();

We can note, therefore, that the announcements of data prefix in NDN networks is performed as the same manner as in IP architecture. The unique difference between them consists on the IP network announces being a structured IP prefix with 4 bytes, while announces in NDN networks have an arbitrary data prefix size.

3.7 NDN Routing Protocols: a survey

3.7.1 OSPFN: OSPF for Named Data Networking

OSPFN [35] is an extension of the known IP Open Shortest Path First (OSPF). It still uses IP as routers IDs, relies on GRE (Generic Routine Encapsulation) tunnels to cross legacy network and computes only a single best next-hop for each name prefix. It installs the best next-hop to each name prefix in the FIB. In order to carry name prefixes in routing messages, OSPFN defines a new type of Opaque Link State Advertisement (OLSA). OLSA allows for information to be announced in the network. Although legacy nodes will not use these LSAs to build their topology, they will still forward them, which ensures backward compatibility. Open source routing suites, such as Quagga OSPF, provide an API that allows easy injection and retrieval of OLSAs through OSPF, which makes OLSA a perfect candidate for advertising name prefixes [35]. Each NDN node runs CCND (Content Centric Network Daemon), OSPFN and OSPFD (OSPF Daemon) in parallel. OSPFN builds name OLSAs and injects them into the local OSPFD, which floods OLSAs to the entire network. When OSPFD at a node receives an OSLA, it delivers the OLSA to its local OSPFN. OSPFN can obtain the router ID of a name OLSA and query OSPFD to retrieve the next hop to the router because each LSA carries the ID of the router that originated the LSA. Furthermore, OSPFD still floods its regular LSAs and computes the shortest path tree based on the topology. OSPFN then installs the name prefix and its associated next hop in the CCND FIB. Figure 3.1 illustrates the relationship between the three protocols.

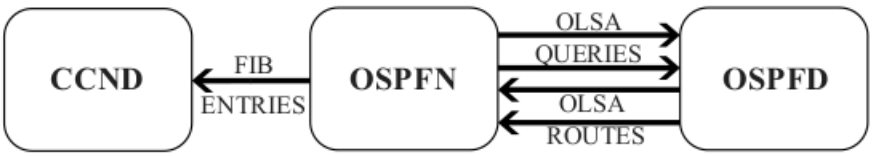


Figure 3.1: Relationship between CCND, OSPFN and OSPFD, from [35].

It should be noted that, although OSPF can build a FIB with name prefixes, it has important limitations. One important limitation affects the forwarding strategy, which cannot explore natively multiple paths to retrieve a piece of named data. Consequently, OSPFN limits the effectiveness of

the NDN. In order to allow multi-path routing support over OSPFN, the operators may manually configure a list of alternative next-hops and install them in the FIB in addition to the best outgoing interface.

OSPFN is the first routing protocol developed for NDN network, it is a short-term solution to get the NDN testbed operational; it does not support dynamic multipath forwarding and has no mechanism to authenticate routing data [25].

3.7.2 NLSR: NDN Link State Routing

NDN Link State Routing protocol (NLSR) [36, 37], as its name indicates, is a link state routing protocol like OSPFN, developed to replace OSPFN inability. It is an NDN intra-domain routing protocol, that uses names instead of IP addresses to identify the various components of a routing system such as network, routers, processes, data, and key. Furthermore, it uses the packets Interest and Data to distribute routing updates. It can use any underlying communication mechanisms that NDN uses (Ethernet, IP tunnels, TCP/UDP tunnels) for routing message exchanges. NLSR uses hierarchically structured names scheme to identify various components: routers, routing processes, routing data, and keys because the relationship among them is inherently hierarchical.

- The routers are named as$\Rightarrow$ $/network/site/router$
- The NLSR process $\Rightarrow$ $/networksite/router/NLSR$
- The NLSR data $\Rightarrow$ $/network/site/router/NLSR/LSA/site/router/type/version$
- The routers share the same $/network/siteprefix$ that means they belong to the same site, and if they share $/network/$ that means they belong to the same network.

NLSR uses Link State Advertisements (LSAs) to disseminates routing information. It uses two kinds of LSAs, adjacency LSAs and Prefix LSAs. NLSR uses adjacency LSAs for building a network topology and Prefix LSAs for distributing name prefix reachability, Figure 3.2 shows the format of LSA. The latest versions of the LSAs are stored in a Link State Database (LSDB) at each router. In order to establish and maintain adjacency relations between neighbor routers, every router periodically sends a digest of their LSDB to other routers in the network using NDN Interest packets. When a router produces a new LSA and its digest changes, it will reply to the Interest with the name of the new LSA and other routers can then fetch the new LSA data. To detect the failure or recovery of any of its links or neighbor processes, each NLSR router sends periodic INFO Interest messages to each neighboring router. If an INFO Interest times out, NLSR will resend INFO Interests in case the Interest was lost. If there is no response from the neighbor after a few tries, the adjacency with the neighbor is considered INACTIVE. Furthermore, a new adjacency LSA for the change will be send to the entire network. The NLSR process will continue to send INFO Interests

to its neighbors, when the adjacency recovers, NLSR will receive a response to its INFO Interest and set the adjacency status to ACTIVE and it will send again a new adjacency LSA will send to the entire network about the topology change. In addition, it advertises name prefixes from both static configuration and dynamic registration by local content producers. Whenever any name prefix is added or deleted, it also disseminates a new prefix LSA. The latest version of the LSAs are stored in a Link State Database (LSDB) at each node. Figure 3.4 show LSA Dissemination via ChronoSyn, which is used in the new NLSR.

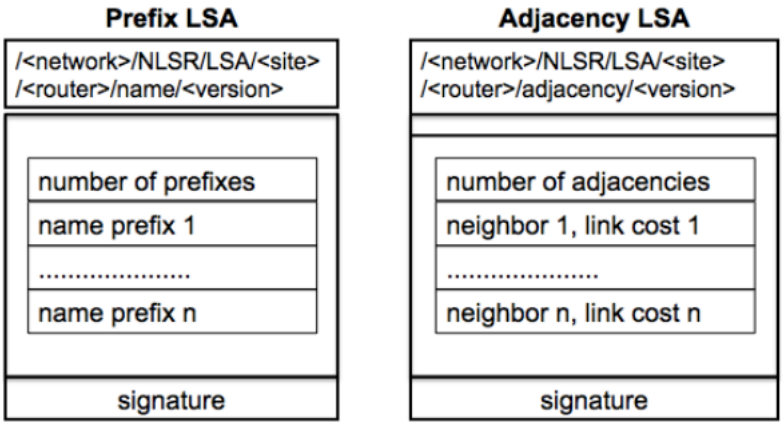


Figure 3.2: LSA format from [37].

The initial NLSR design [36], which was developed in 2013 uses CCNx's SYNC/repo mechanism to distribute LSAs between routers. However, during the extensive test of NLSR over an internal testbed of 12 nodes, several problems were discovered with CCNx's SYNC/repo implementations, that consist on high memory consumption, inability to delete information from the repo, and failure to notify NLSR of routing changes when the update rate is high [25], [38].

The new NLSR is redesigned and reimplemented to work on the new platform developed in 2014 [39] with new forwarder NFD (NDN Forwarding Daemon) [40] and libraries (ndn-cxx). Furthermore a new mechanism to distribute LSAs called ChronoSync [41] is used to replace CCNx's SYNC/repo mechanism.

Concerning security, NLSR uses also a hierarchical trust model for intra-domain routing. Each Data packet is digitally signed by the originating router, with the signature covering the name, the content and the metadata which is useful in signature verification. The receiver can check the signature by using the key locator carried by data packet, which is one piece of the metadata, that indicates the name of the key used to sign the packet. In addition, the receiver needs to verify that the key is trusted to sign the LSA, which requires a trust model for key authentication.

In NLSR routing protocol trust management is modeled as a five-level hierarchy reflecting the administrative structure of an intra-domain routing protocol, as shown in Figure 3.3 and Table 1

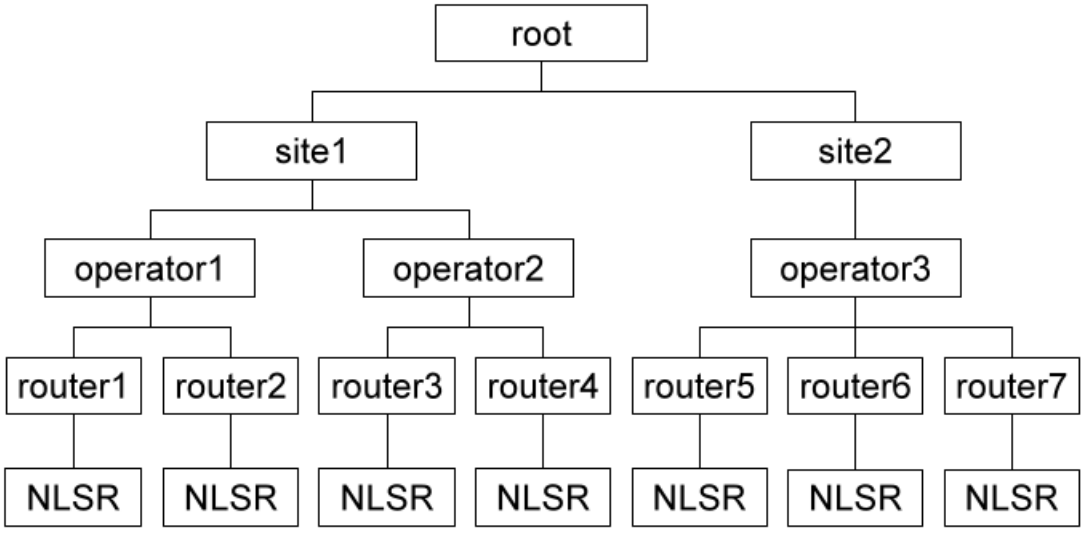


Figure 3.3: NLSR trust hierarchy, from [37].

which show respectively NLSR trust hierarchy and key names. So, at the top level (root), there is a single authority that is responsible for the whole network. The network consists of multiple sites, and each site has one or more operators who collectively manage the routers of the site. Moreover each router can create a NLSR process that produces LSAs.

Key Owner	**Key Name**
Network	/<network> /KEY/ <key>
Site	/<network> / <site> /KEY/ <key>
Operateur	/<network> / <site> / <operateur> /KEY/ <key>
Routers	/<network> / <site> / <router> /KEY/ <key>
NLSR	/<network> / <site> / <router>/NLSR/KEY/<key>

Table 3.1: Key names, from [37].

The NLSR trust model allows to establish a chain of keys to authenticate LSAs. As result, an LSA must be signed by a valid NLSR process running on the same router where the LSA originates. To become a valid NLSR process, the process key must be signed by the corresponding router key, which in turn should be signed by one of the operators of the same site. Each key of site operator must be signed by the site key, which must be certified by the network authority using the network key, that called trust anchor [38].

In order to support multi-path routing, NLSR uses a modified Dijkstra's algorithm to get a ranked list for each name prefix in the FIB table, which can be used by forwarding Strategy of NDN. It is worth mentioning that, the new NLSR [42] has been deployed on the inter-continental NDN testbed since August 2014 [38], and many bugs in the code have been detected. As result,

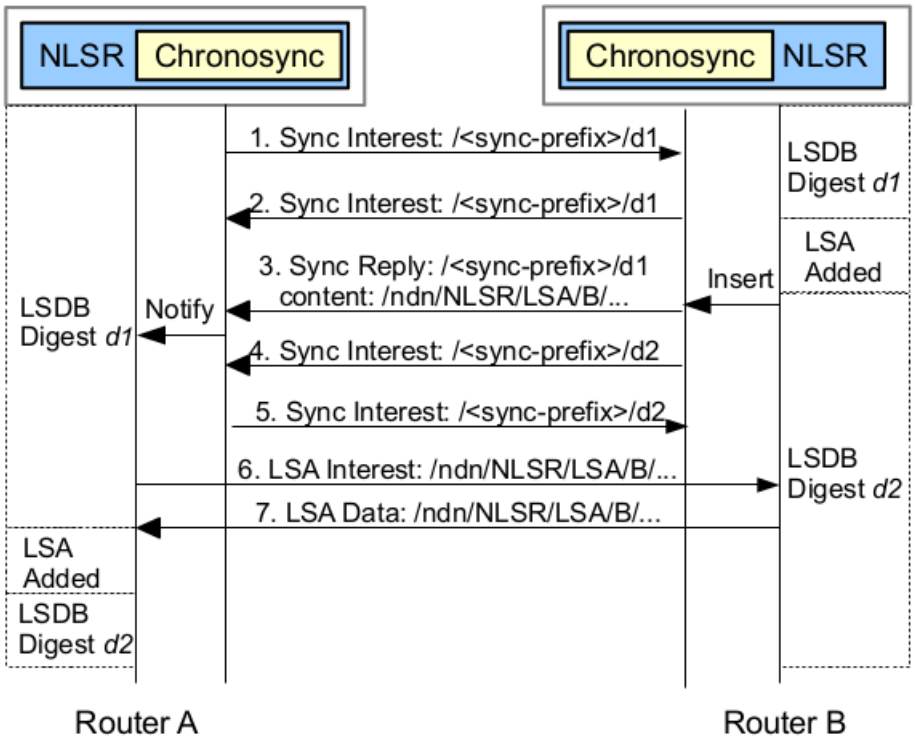


Figure 3.4: LSA Dissemination via ChronoSync, from [37].

a new NLSR 0.2.0 on April 30, 2015 was released, with improvements of such bugs. The last released version was NLSR 0.2.1 dated from June 30, 2015, which was updated with last versions of platform ndn-cxx and NFD, and the corrections bugs that were detected between April 30 and June 30, 2015.

3.7.3 Hyperbolic routing

The Hyperbolic routing is a geometric routing scheme that relies on hyperbolic coordinates to send packets efficiently through a network. In [43], [44], it is described,how hyperbolic metric spaces underlying complex networks enable efficient greedy forwarding without any global knowledge of the network topology is described. Assuming that a mechanism for retrieving the coordinates of a name prefix exists, each Interest can carry the coordinates of the name prefix and routers and use greedy routing to forward the Interest, i.e., choose the next hop(s) for the Interest based on the distances between a router's neighbors and the desired name prefix. This scheme is highly scalable as there is no need to maintain a routing table or FIB and there are no dynamic routing updates except to learn the coordinates of neighbors.

Hyperbolic routing is adapted to address the routing scalability problem in NDN, for supporting unbounded name space and also a large Internet topology. This idea is supported by [45], since

hyperbolic routing uses pre-assigned hyperbolic coordinates to direct interests toward data and it does not use traditional routing tables or exchange routing updates upon changes in the network topology.

However, it introduces potential drawbacks of sub-optimal routes for some destinations. To overcome these drawbacks, a new forwarding strategy has been designed called Adaptive Smoothed RTT-based Forwarding (ASF) and have been evaluating the performance of hyperbolic routing with ASF [45].

In [46], it has been applied an hyperbolic greedy routing on the current NDN testbed topology, where the calculation shows a high success ratio of packet delivery. In [25], [47], a basic hyperbolic routing in NLSR has been implemented to disseminate hyperbolic coordinates in link state announcements. The methodology to embed a router topology for a given NDN testbed site is developed into the hyperbolic plane and a simulation of the NDN networks growth by using hyperbolic coordinates is done.

In [48], an updated NDN testbed topology is embedded into the hyperbolic plane using more recent Internet topology data, and a new, simpler and more efficient, hyperbolic network mapping algorithm is developed. Furthermore, a basic hyperbolic routing is implemented in NLSR by disseminating hyperbolic coordinates in link state announcements. In order to evaluate the feasibility of hyperbolic routing in NDN, experiments have been performed on Emulab by comparing it with link-state routing under various conditions. Furthermore, a stand-alone software package for hyperbolic network generation and greedy routing simulation has been developed and released.

In [49] in order to confirm that hyperbolic routing is a feasible candidate for inter-domain routing of NDN, its performances have been evaluated and compared to the link-state routing in a large topology. The comparison criteria were: the packet loss ratio, RTT, message overhead, and failure response time of data retrieval. The experiments are conducted with various forwarding strategies, failure conditions, and topologies. The obtained results conclude that the hyperbolic routing must have a forwarding overhead much lower than that of smart flooding strategy to be a viable solution for NDN.

The performance evaluation in [45] demonstrate that the forwarding plane HR overhead in such networks is much lower than control overhead of NLSR, while achieving similar packet delay and loss rates. Furthermore, when the size of network increase, per-node message overhead of HR is almost constant and close to zero, compared to the usual polynomial growth experienced by traditional links-state routing.

There are several technical issues that require further investigation in order to include mechanisms to compute and distribute hyperbolic coordinates in a decentralized manner, properly taking into account link delays [45].

3.8 QoS based routing in NDN network

The Internet is characterized by the heterogeneity of its links and the dynamic traffic conditions. This requires taking into account the quality of service (QoS) in network and especially in routing to react to the requirements of users. The goal of QoS based routing is to find the best network path according to the precise criteria of the desired quality of service (delay, loss rate, bandwidth, etc.) which satisfies the given constraints and simultaneously optimizes the utilization of resources [50]. The integration of QoS parameters increases the complexity of routing algorithms. In other words, the problem of determining the path that satisfies QoS constraints (two or more QoS parameters non-correlated over one path, for example, delay and cost) is known to be NP-Complet [51]. The QoS based routing issue has been widely discussed in the literature for the TCP/IP architecture; while it has just started in the ICN architectures.

3.8.1 Problem Formulation

We formulate the routing with QoS in NDN network as follow. Let $G = (X, U)$ be the graph consisting of a set $X = (x_1, x_2, ..., x_n)$ with $|X| = N$ nodes, and a set $U = (u_1, u_2, ..., u_n)$ with $|U| = M$ links. The nodes represent routers, consumers or providers, while the links represent the communication links. A specific link of the set U between the nodes u and v is denoted by (u, v). Each link (u, v) of U is characterized by an m-dimensional vector W such as $W(u, v) = [w_1(u, v),\ w_2(u, v),\ ...\ ,\ w_m(u, v)]$. Where $W_i(u, v) \geq 0$ if (u, v) belongs to U, and m components refer to the QoS criteria such as $delay$ and $cost$ provided by the link (u, v). Finally, let $L(u, v) = [L_1(u, v),\ L_2(u, v),\ ...,\ L_m(u, v)]$. Where $L_i(u, v) \geq 0$, is the vector of the QoS constraints set by the user (or the application).

The consumer expresses the desired data by sending in the network an Interest packet that contains the name of the data along with quality of service constraints. The requested data, if exists, can be found at the producer node, a repository or in-network cache. Therefore, when a copy of the requested data is found, it flows back along the same path traversed by the Interest but in the opposite direction. A path in the topology G is denoted $P(Cons, X)$. It connects the consumer node (denoted $Cons$), who requested data and X, which is the node that has a copy of the data requested by the consumer. The QoS routing algorithms allow calculating the path P that optimizes one or more QoS constraints. The main idea of our work is to collect in real-time different dynamic QoS parameters, and exploit them to guide different user data requests toward potential sources of data via the best path P, and to redirect the founded data back to consumers with the required quality of service. To achieve this objective, we proceed by steps. First, we monitor the links of network to estimate the QoS parameters. Second, these estimations are collected by routers. Finally, routers exploit the collected information to determine the paths that allow the satisfaction of the quality of service constraints required by the user (or the application).

As explained in Sections 3.3, 3.4 and 3.5, it is clear that the three tasks mentioned previously belong to the forwarding plane. This latter, which is two-way traffic, allows us to observe the forwarding performance (and the retrieved data), to estimate the QoS parameters of links, and to collect and records this information in the routers FIB table entries. This information called forwarding plane information along with the routing plane information (recorded too in FIB table entries) are used as input for our forwarding strategy, which determines the paths that allow the satisfaction of the quality of service constraints.

Based on the information introduced in this formulation, the solution that we propose for resolving the QoS based routing issue, throughout this dissertation, consists on the designing and implementation of new forwarding strategies that are enable to collect, in real-time, QoS parameters measurements of network links and integrate them into the different decisions taken for forwarding the user requests.

3.9 Forwarding Strategies: a Survey

As explained previously in Sections 3.3, 3.4 and 3.5, in NDN networks, the forwarding plane is responsible for the forwarding of the packets, while the routing plane helps the forwarding plane and builds the FIB table.

As a fundamental issue in ICN, forwarding strategies constitute a research topic that has attracted much attention in recent years. In NDN network, the routing process success depends essentially on the forwarding strategy, which is the key module for the NDN node architecture.

3.9.1 High level taxonomy

There are a lot of forwarding strategies proposed in the literature to address many NDN issues. We propose to classify them according to the issue that the forwarding strategy is designed to handle as depicted in the Figure 3.5.

3.9.1.1 Forwarding strategies that handle the network failure

The authors in [52] proposed On-demand Multi-Path Interest Forwarding (OMP-IF) strategy which starts by identifying a set of paths based on the disjointness of paths to content locations by using a single face per data name prefix. Then, it distributes (split) Interests simultaneously over discovered paths, which is triggered by the client, based on the characteristics of paths (delay). The authors in [53] propose SAF that imitates a self-adjusting water pipe system, which guides and distributes intelligently Interests through network crossings circumventing link failures and bottlenecks. SAF is implemented and evaluated by conducting extensive simulations using the ndnSIM for ns3 simulator.

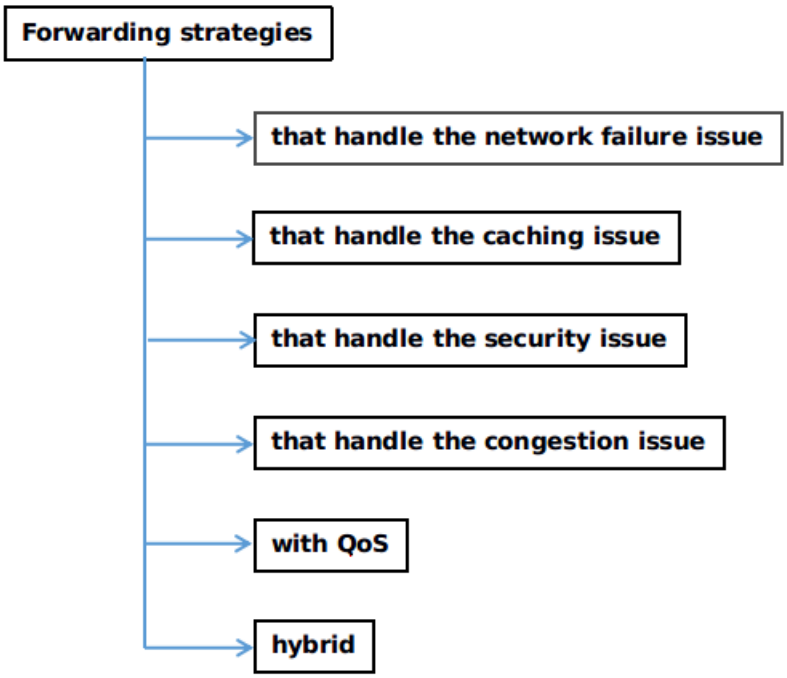


Figure 3.5: High level forwarding strategy taxonomy.

3.9.1.2 Forwarding strategies that handle the caching issue

The literature forwarding strategies that address the caching issue are the following. Chiocchetti et al. [54] propose *INFORM*, a distributed on-line request forwarding algorithm based on Q-routing. *INFORM* discovers temporary copies of not addressed contents in routing tables and forwards requests over the best performing interface at every hop. Authors in [55] compared two approaches exploitation and exploration: a deterministic exploitation of forwarding information towards a known copy, and a random network exploration towards an unknown copy, via request flooding. They suggest a hybrid solution: trade-offs of exploitation and exploration approaches. The authors in [56] propose Nearest Replica Routing, which couples caching and forwarding. This approach extends aNET [57], that approximates the behavior of multi-cache networks by leveraging existing approximation algorithms for isolated caches. The iterative algorithm proposed in [56] uses an oracle providing information on the availability of content in all caches in the network. The strategy send out the incoming interest toward the face with the shortest distance to the content. So, it prefers nearby caches rather than the content origin. iNRR is implemented in the ccnSIM simulator [58]. It is worth mentioning that the authors indicate that an implementation of a perfect oracle is not feasible in a real environment.

3.9.1.3 Forwarding strategies that handle the security issue

It is possible to developer a forwarding strategies to handle only the security issue. Afanasyev et al. [22] propose a new forwarding strategy that handles the problems of prefix hijack (security) with other problem such as link failure, and congestion.

3.9.1.4 Forwarding strategies that handle the congestion issue

There are no reliable congestion control mechanisms in NDN network that enable to handle multi-path transmission until today. The literature attempts for solving this issue can be classified into two approaches, the end-to-end approach of TCP by adjusting consumer Interest window size using the AIMD algorithm and Hop-by-hop interest shaping mechanisms approach. The literature work that follows the first approach, the end-to-end approach of TCP by adjusting consumer Interest window size using the AIMD algorithm, is the following. The authors in [59] designed an Interest Control Protocol for Content-Centric Networking protocol (ICP) driven by a window-based AIMD controller of the Interest rate expressed at the receiver. The goal of ICP is to control Data packet retrieval from the different routes that constitute each path between user and repository. They provide an analytical characterization of stationary rate of user and queues dynamics, under general traffic demand that demonstrate the impact of CCN transport and caching dynamics on user performance.

In [60] the authors propose CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking. CCTCP uses a novel anticipated interests mechanism, which enables reliable prediction of content chunks location before they are requested and consequently accurately estimates the retransmission timeout.

In [61] the authors propose a congestion control mechanism for multipath communication over content-centric networks. This latter is based on a Remote Adaptive Active Queue Management (RAAQM) at the receiver that performs a per-route control of bottleneck queues along the paths. In [62], the authors carried out an Empirical Study of Receiver-Based AIMD Flow-Control Strategies for CCN. They focused receiver-based timeout-driven AIMD flow-control in CCN, and particularly in relevance to TCP. The result of this study demonstrates that timeout-based flow-control in CCN in general is not compatible to the CCN philosophy and its performance varies depending on the content entropy in the network. Since in NDN network, by means of caching feature, the requested data can be retrieved from multiple providers through multipath, it is very hard to set the Retransmission Timeout (RTO) as in TCP to detect congestion. Consequently, [62], [59] and [60] propose some solutions that tries to predict RTT more accurately by piggybacking location information of future content on the Data packets.

Concerning the literature work that follow the second approach, Hop-by-hop interest shaping mechanism approach are the following. The authors in [63] propose Joint Hop-by-hop and Receiver-Driven Interest Control Protocol for Content-Centric Networks (HR-ICP) to regulate user

requests in terms of Interests either at the receiver and at intermediate nodes via Interest shaping. They demonstrate by using of CCN packets-level simulations that HR-ICP is more effective that the literature mechanism that ensure controlling only at receiver because HR-ICP accelerates congestion reaction by reducing the loss rate.

The authors in [64] propose the first hop-by-hop Interest shaping congestion control mechanism for CCN communications called HoBHIS. This latter is designed to avoid the congestion that can occur in the output interface of a CCN node. The main idea of HoBHIS consist on monitoring the transmission buffers of CCN router interfaces to compute the appropriate Interests rate. The performance of HoBHIS is evaluated under ns2 simulator.

Wang et al. [65] and [66] propose a hop-by-hop Interest shaping algorithm that takes into account the unique interdependence between Interests and data in NDN to achieves proportional fairness between two-way traffic in order to use efficiently the available bandwidth and avoid congestion. They also proposed a signaling mechanism to periodically and reliably notify the routers on downstream and end users of the congestion.

The authors in [67] and [68] tackle the problem of joint multipath congestion control and request forwarding in ICN. They are formulated as a global optimization problem with the twofold objective of maximizing user throughput and minimizing overall network cost. They decompose the problem into two subproblems for congestion control and request forwarding. Then they derive a family of optimal congestion control strategies at the receiver and a set of optimal dynamic request forwarding strategies. They also design a distributed algorithm to be implemented by ICN nodes in conjunction with the proposed receiver-driven congestion control. The proposed solution enable to achieve optimal multipath congestion control, but its objective is to minimize the number of pending Interests on the most loaded interface. Consequently, the incoming Interests may not be forwarded along the best routing paths even if they are not congested.

The authors in [69] propose a Practical Congestion Control Scheme for NDN. In order to design this proposed congestion control mechanism called PCON, the hop-by-hop stateful forwarding plane of NDN and the congestion estimation of CoDel AQM are adopted. Thus, based on the CoDel AQM algorithm, PCON detects congestion. Then it marks certain packets for signaling the congestion for consumers. So that, the downstream routers direct the traffic towards alternative paths and consumers can mitigate their Interest sending rates.

3.9.1.5 Forwarding strategies with QoS

The most known forwarding strategies are integrated in the NDN simulator [70], [71]. In [70] a color-coding (Green, Yellow and Red) is used to represent the interface status: (Green) for the interfaces that return data, (Yellow) when interface may or may not return data, and (Red) where the interface does not work. The simplest approach is the flooding strategy that forwards incoming Interest to all available interfaces in FIB entry for the prefix of Interest, except the incoming inter-

face of Interest. The best route strategy forwards the received Interests to the highest-ranked green interface. If there is no available green interface, it will be forwarded via the highest-ranked yellow interface. For the smart flooding strategy, the Interest is forwarded only to the green highest-ranked available interface; otherwise, all yellow interfaces will be used to forward the Interest.

In [71], the Broadcast strategy forwards the received Interests to all upstream interfaces; the Client Control Strategy allows a local consumer application to choose the outgoing interface of each sent Interest; the Best Route strategy forwards the incoming Interests to the upstream interface with the lowest routing cost and the NCC strategy is a re-implementation of the CCNx 0.7.2 default strategy.

In [72], [73], the authors propose a Greedy Ant Colony Forwarding (GACF) algorithm which uses the ISP-based aggregation to reduce the content naming space. They used two kinds of ants, the first one to explore the network path and the second to get the data and reinforce the optimization. The idea was inspired form AntNet [74]. This approach suffers from the big traffic generated only for the exploration.

In [75], the authors proposed Ant Colony Optimization-inspired Routing with Content and Similarity Relation (AIRCS), where a continuous content concentration model conducts the interest forwarding, and a similarity relation model between two content routers acts as a heuristic factor to facilitate the interest forwarding. After, that a computation scheme uses the results of both modules to determine the outgoing interface. AIRCS is implemented over C++ and the performance evaluation is done on a personal computer, without using a known simulator or test bed.

Authors in [76] present *SoCCeR-Services* over Content-Centric Routing, that extends CCN with integrated support for service routing decisions leveraging ant-colony optimization. The authors in [77] propose a probabilistic ant-routing mechanism that enables multipath transmissions for CCN nodes. In [78], the authors propose probability-based adaptive forwarding strategy (PAF) in NDN by customizing ant colony optimization and focuses on delay minimization.

The authors in [79] proposed three forwarding strategies that work at the client side: Lowest Cost Strategy(LCS), Multiple Attribute Decision Making Strategy and Selective Parallel strategy(SP). Those three strategies use three estimators modules to compute respectively the following QoS metrics: delay, packet loss, and bandwidth. The Lowest Cost Strategy forwards interest requests via the first interface that satisfies a hard-coded threshold. In MADM strategy two thresholds values are used as a boundary of acceptable values. It uses the three measurements of estimators as input to choose the first matching interface that satisfies certain boundary values, with regards to bandwidth, delay or loss. The main idea of the SP strategy consists on forwarding interest packets on multiple interfaces simultaneously when there is no interface matching the condition then it.

In [80] the authors propose Fast Pipeline Filling (FPF). The FPF strategy starts by identifying the set of interfaces whose number of pending interest is lower than the related link capacity for the incoming Interest. Then, among this set of interfaces, it selects and forwards the arriving Interest

via the interface that has the lowest Round Trip Time (RTT).

The authors in [81] propose adaptation of multi-path forwarding strategies for Content Centric Networking (CCN) to operate under mobility. This latter is a weighted round robin scheme among interfaces whose weights are inversely proportional to the interface round trip time.

The authors in [82] propose three forwarding strategies for evaluating CCN multi-path Interest. The uniform strategy, which forwards the Interests towards an interface that is selected randomly. The round robin strategy, which forwards the Interests towards available node interfaces in equally manner. The parallel strategy, which broadcasts the interests towards all available node interfaces.

A more recent work [83] uses the ant colony algorithms for obtaining QoS. This strategy try to satisfy two constraints (maximum delay and minimum bandwidth), so, among the interfaces that satisfy these constraints, it selects the best interface in terms of probability that is computed by a special formula, that takes the cost and delay pheromone, which is computed separately. The impact of each parameter for the pheromone update (cost, delay) is weighted separately.

3.9.1.6 Hybrid forwarding strategies

This class regroups forwarding strategies that handle two or more different kind of issue at the same time.

Afanasyev et al. [22] propose a new forwarding strategy that handles the problems of prefix hijack (security), link failure, and congestion. The main idea is to use the Interest *Nack* as a tool for the first time, and color-coding (Green, Yellow, Red) to represent the working status of each interface.

The authors in [23] propose two congestion control mechanisms for NDN network with adaptive forwarding, a simple Interest limiting (SIL) mechanism which, combined with adaptive forwarding, is able to achieve hop-by-hop multipath congestion control and Dynamic Interest Limiting (DIL) mechanism which is more practical. The idea used for designing these mechanisms consists on enforcing an Interest Limit on each interface, which specifies the maximum number of pending Interests allowed on this interface.

Bouacherine et al. [84] propose Parallel Multi-Path Forwarding Strategy (PMP-FS) which proactively splits traffic by determining how the multiple routes will be used. It takes into consideration NDN in-network caching and NDN Interest aggregation features to achieve weighted alpha fairness among different flows.

3.10 Conclusion

In this chapter, we have introduced the difference between the routing in IP and NDN networks, and we have discussed the features of routing in NDN. We have detailed the role of routing and forwarding plane, and the advantages of coordination between them. Furthermore, we have presented

the works concerning NDN routing and forwarding that we classify according to the handled issue. In addition, we have evoked the QoS based routing, and we have introduced the problem formulation. Finally, we have demonstrated why the solution that we propose for this issue consists on a new forwarding strategies with QoS. In the next chapter, we will present our first contribution, which consist on a new forwarding strategy with QoS for routing in NDN.

Chapter 4

QoS-based Forwarding Strategy for NDN

Contents

4.1 Introduction 69
4.2 Reinforcement Learning 70
4.2.1 Mathematical Model of Reinforcement Learning 71
4.2.2 Value functions 72
4.2.3 Rewards 73
4.2.4 Definitions of functions of V-value and Q-value 73
4.2.5 Value function and optimal policy 73
4.2.6 Q-Learning 75
4.3 Our proposal: the QoS forwarding strategy 76
4.3.1 Functional Model 76
4.3.2 Designing the QoS forwarding strategy 78
4.4 Performance Evaluation 88
4.5 Conclusion 91

4.1 Introduction

The Internet is characterized by the heterogeneity of its links and the dynamic traffic conditions. This requires taking into account the quality of service in network and especially in routing to react to the requirements of users. This issue has been widely discussed in the literature for the IP

architecture; while it has just started in the ICN architectures. In this chapter, we study the QoS-based routing in NDN. We show that the QoS-based routing could be ensured by considering our proposed forwarding strategy.

This chapter presents the design of **QoS-FS**, a new adaptive forwarding strategy for NDN with quality of service (QoS). At each node of the network, **QoS-FS** monitors, in real-time, ingoing and outgoing networks' link to estimate the QoS parameters and integrate them into the different decisions taken to determine when and which interface to use to forward an Interest. Therefore, making forwarding decision adaptive to network conditions and preferences of user.

The rest of this chapter is organized as follows: the reinforcement learning used in the design of the proposed forwarding strategy is introduced in Section 4.2, our proposal is presented in Section 4.3 and evaluated in Section 4.4. Finally, we conclude the chapter in Section 4.5 .

4.2 Reinforcement Learning

Learning is a process that aims to improve the performance of a system on the basis of past experiences. This improvement depends initially on the ability to extract information from these experiments and, secondly, on the use of this information in order to improve a certain defined performance function. According to the nature of the available information and the goal, three learning families can be distinguished: supervised learning, unsupervised learning and reinforcement learning.

Supervised learning is an automatic learning technique where one seeks to automatically produce rules from a learning database containing "examples" (usually cases already treated and validated). It is the learning mode most commonly used. The unsupervised learning is an automatic learning method, where it aims to divide a heterogeneous group of data into subgroups. Since only input information are available, it must therefore determine its outputs based on the similarities detected between the different inputs, by using a self-organizing rule.

There are many problems that are difficult to solve by using the previous kinds of learning. In the case where there is only qualitative information available to evaluate the calculated response, reinforcement learning uses this assessment to improve system performance. This form of learning is a method of learning by trial and error.

In reinforcement learning, only a qualitative measure of error is available [85] [86]. In this case, the agent receives stimuli from the environment and reacts by choosing an appropriate action for his behavior. The agent reaction is then judged against a predefined objective in the form of a "reward" rating. The agent receives this "reward" and must integrate it, for modifying his future actions and thus achieve optimal behavior. An action leading to a negative rating will, in the future, and under the same conditions, be used less than a positively rated action.

The reinforcement learning results from two simple principles:

- For a given state, if an action immediately causes something bad, then the system learns to not do this action when it is in this state.
- For a given state, if all possible actions lead to something bad, then the system will learn to avoid being in this state.

The reinforcement learning consists of learning for an autonomous agent a behavior to adopt when interacting with its environment, in order to achieve objectives without any external intervention. Figure 4.1 illustrates the principle of reinforcement learning.

Figure 4.1: Reinforcement Learning model.

4.2.1 Mathematical Model of Reinforcement Learning

Consider a system defined by three states, for which each state can perform two actions possible a_1 and a_2 (Figure 4.2) [87]. The transition from one state to another is carried out according to the probability P_{21}^{a1}, P_{21}^{a2}, leading to a reward signal respectively r_{23}^{a1} and r_{23}^{a1}.

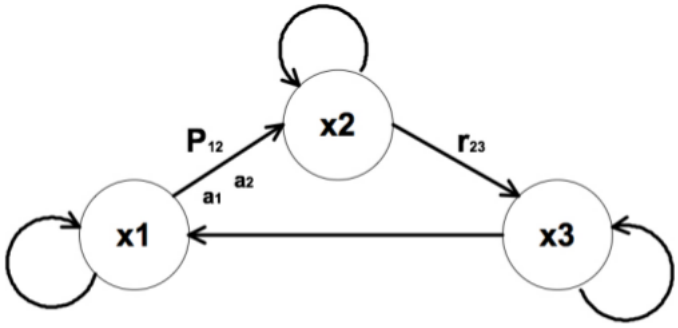


Figure 4.2: Example of Markov process with 3 states and 2 actions.

There are some notations:

- X: the states set of the system and $x_t \in X$ is a state of the system at time t.
- A_x: the set of possible actions of state x.

- $A = \cup_{x \in X} A_x$: set of all actions.
- $a_t \in A_x$: the action applied at time t the state x_t.
- $P_{xy}a = Pr(x_{t+1} = y | x_t = x, a_t = a)$: defines the probability of transition from state x to state y under action a.
- R: all the rewards that the system can receive from the environment, such as:

$$r : (X, X, A) \rightarrow R(x_t, y_t, a_t) \rightarrow r_{x_t y_t}^{a_t} \tag{4.2.1}$$

  Where r is the reward return from the environment by applying the action a_t during the transition of state $x_t \rightarrow y_t$.

- π: the policy adopted, that chooses the action a in the state $x : a = \pi(x)$, such as:

$$\pi : X \rightarrow Ax \rightarrow a \tag{4.2.2}$$

- Π: the set of policies (called control laws), such as:

$$V^{\pi} : (X, \Pi) \rightarrow R(x, \pi) \rightarrow V^{\pi}(x) \tag{4.2.3}$$

- V^{π}: the measure of the quality of policy π of state x.
- U: the set of the couples action-state.
- Q:the value function of couple action-action, such as:

$$Q^{\pi} : (U, \Pi) \rightarrow R(x_0, a, \pi) \rightarrow Q^{\pi}(x_0, a) \tag{4.2.4}$$

The function $Q^{\pi}(x_0, a)$ measures the quality of the policy π when the system is in the state x_0 and the action a is chosen. The definition of the functions Q and V (Q-Value and V-Value) is fundamental because the optimal policy will be that which will lead to the best quality (V^* and Q^*), defined by:

$$\pi^* = arg \max_{a \in A_x} Q^*(x_0, a) \tag{4.2.5}$$

The optimal policy is that which maximizes the criteria V^{π}. The algorithms of reinforcement learning thus aim to maximize in an iterative way the function π.

4.2.2 Value functions

In this section, we present the relationship between value functions and decision policies.

4.2.3 Rewards

A reward at a given time r_t is the signal received by the agent during the transition from the state x_t to the state x_{t+1}, during the execution of the action a_t, defined by: $r_t = r^a_{xy}$.

The global reward represents the sum of all punctual rewards. It is used to evaluate the behavior of the system over the long term. Where a test is finished, this sum is written as follow:

$$R_{et} = \sum_{k=0}^{T} r_{t+k+1} \tag{4.2.6}$$

We define a sum of damped-off punctual rewards by:

$$R_{et} = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{4.2.7}$$

where $0 < \gamma < 1$ and T are the end of the test.

4.2.4 Definitions of functions of V-value and Q-value

The value function, V-value, represents the sum of punctual rewards by using the policy π. The objective of this function is to assess the quality of the states under a given policy. It is defined as follows:

$$V^{\pi}(x) = E_{\pi}[R_{et}|x_t = x] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|x_t = x] \tag{4.2.8}$$

In the same way, a function Q-value, denoted, $Q^{\pi}(x, a)$ represents the expectation of the global reward when the system realizes the action a under the policy π, such as:

$$Q^{\pi}(x, a) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|x_t = x, a_t = a] \tag{4.2.9}$$

The estimation of these value functions is necessary in most reinforcement learning algorithms. These value functions can be estimated from experiments [87], [88].

4.2.5 Value function and optimal policy

A policy $\pi = \pi^*$ is optimal if $V^* = V^{\pi^*}(x) \geq V^{\pi^i}$ with $\forall\ x \in X$ and for all policies π^i [88]. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expectation of global reward:

$$\pi^* = arg\max_{\pi} V^{\pi}(x) = arg\max_{\pi} E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|x_t = x] \tag{4.2.10}$$

The equation (4.2.8) can be expressed in the form:

$$V^{\pi}(x) = \sum_{y \in X} P_{xy}^{\pi(x)}[r_{xy}^{\pi(x)} + \gamma^{V^{\pi}}(y)] \tag{4.2.11}$$

This equation means that the utility of a state under policy π is an expectation over all available states of the sum of the global reward received by applying $\pi(x)$ and the value of y under the policy π. This equation also applies to all policies, in particular to optimal policy π^*:

$$V^*(x) = V^{\pi^*} = \sum_{y \in X} P_{xy}^{\pi^*(x)}[r_{xy}^{\pi^*(x)} + \gamma^{V\pi^*}(y)] \tag{4.2.12}$$

By definition, all optimal policies have the same value function V^*. Therefore, the action chosen by the optimal policy must maximize the following expression:

$$\sum_{y \in X} P_{xy}^{a}[r_{xy}^{a} + \gamma^{V^*}(y)] \tag{4.2.13}$$

which implies:

$$V^*(x) = \max_{a \in A_x} \sum_{y \in X} P_{xy}^{a}[r_{xy}^{a} + \gamma^{V^*}(y)] \tag{4.2.14}$$

Finally, the optimal policy can be determined as follows:

$$\pi^* = arg \max_{a \in A_x} \sum_{y \in X} P_{xy}^{a}[r_{xy}^{a} + \gamma^{V}(y)] \tag{4.2.15}$$

These equations assume knowledge of the system model(r_{xy}^{a}, P_{xy}^{a}). However, in the case where this model is unknown, the Q-value function solves this problem.

The expression $Q^{\pi}(x, a)$ can be put in the following recurring form:

$$Q^{\pi}(x, a) = \sum_{y \in X} P_{xy}^{a}[r_{xy}^{a} + \gamma^{V^{\pi}}(y)] \tag{4.2.16}$$

The function $Q^* = Q^{\pi^*}$ corresponds to the optimal policy, without necessarily choosing the action $\pi(x)$ by policy π. Finally, the expression of V^* can be written in the form (optimal equation of Bellman):

$$V^* = \max_{a \in X}[Q^*(x, a)] \tag{4.2.17}$$

and the expression of $\pi^*(x)$ becomes:

$$\pi^*(x) = arg \max_{a \in X}[Q^*(x, a)] \tag{4.2.18}$$

These last two equations can be evaluated without knowledge of the system model. Therefore, the problem of RL is to learn the Q-values and V-values. In other words, it is to estimate it in the absence of prior knowledge of the model. One of the efficient methods to solve this problem is the Q-Learning we present in the next section.

4.2.6 Q-Learning

Several algorithms have been developed to find the optimal policy, whether or not using the system model. The main reinforcement learning methods are dynamic programming, TD-Learning, Q-Learning. The dynamic programming is a set of methods for finding optimal policies in the case of a perfect knowledge of the model of the environment [86] [87], as in the Markov Decision Processes (MDP). It allows to find the value function resulting from a given policy or, the optimal policy and the corresponding value function. The TD-Learning method [86] [87] [89] is used for the treatment of long-term problems during prediction. The objective is to predict an event at time T from the instants $t_0, t_1, t_2, ..., t_{T-1}$.

The method of Q-learning [90] is a learning method that does not require a model, and its objective is to find an approximation of the Q-function, denoted Q, which approaches Q^* [91] [92].

$$V^* = \max_{a \in X}[Q^*(x, a)] \tag{4.2.19}$$

The Q-learning algorithm can be summarized in Algorithm 1.

Algorithm 1: Q-learning algorithm.

1 Input: $t = 0$, x: an initial state
2 **repeat**
3 Adjust Q, such as:
4 $Q(x, a) = Q(x, a) + \alpha[r^a_{xy} + \gamma \max_{b \in A_y} Q(y, b) - Q(x, a)]$
5 $x = y$
6 **until** $Q = Q^*$

Watkins in [90] has proved that the values of $Q(x, a)$ reach the optimal values of $Q^*(x, a)$ with the probability of 1.

The Q-learning method has obvious advantages. First of all, it requires no prior knowledge of the system. Then, the vision of agent is very local, which corresponds well to the problem of dynamic routing where each router must decide the path from which it sends its packets without knowing the state of all other routers. In addition, the implementation requires only a Q-value table comparable to a routing table. This latter is updated after each action performed. However, this algorithm has a major disadvantage that is generally found in all reinforcement learning methods: it is the guarantee of the convergence of Q to Q^*, for which strong assumptions are allowed: the process must be Markovian and stationary, and the set of state-action pairs (x, a) must be visited an infinite number of times. This latter hypothesis is not crucial in all subjects of reinforcement learning.

The first attempt of applying the Q-learning algorithm to adaptive routing protocol, called Q-Routing, was proposed in [93]. In Q-Routing each node uses Q-table to store the routing information, Q-values, that represent the state of network. These latter are learned by using Q-learning and the node uses these values to make control decisions. Thus, in Q-routing, the goal is to learn

an optimal routing policy for the network. The Q-values represent the state s in the optimization problem of routing process. The node x uses its Q-tables Q_x to represent its own view of the state of network. The action a at node x is used to choose neighbor y such that it maximizes the obtained Q-value for a packet destined for destination node d. In Q-Routing, each node x has a table of Q-values $Q_x(y, d)$, where $d \in V$ the set of all nodes in the network, and $y \in N(x)$ is the set of all neighbors of node x. The $Q_x(y, d)$ is the best Q-value that a packet would obtain to reach the destination d from node x when sent via the neighbor y. The Q-Routing algorithm is based on exploitation and exploration phases, and the goal of Q-Routing is to balance the exploitation and exploration phase.

- Exploration phase: After receiving the packet $P(s, d)$ sent from node x, destined for node d, node y sends back to x its best estimate $Q_y(\hat{z}, d)$ for the destination d: $Q_y(\hat{z}, d) = \max_{z \in N(y)} Q_y(z, d)$. Using the estimate value, the node x updates its $Q_x(y, d)$ value:

$$Q_x(y, d)^{new} = Q_x(y, d)^{old} + \alpha(Q_x(y, d)^{est} - Q_x(y, d)^{old}) \tag{4.2.20}$$

- Exploitation phase: In this phase, each node stored the Q-values that are used for making locally greedy routing decisions. In other words, when a node x receives a packet $P(s, d)$ destined for node d. Node x looks at the vector $Q_x(*, d)$ in its Q-table and choses his neighbor node $\hat{y}$ so that the $Q_x(\hat{y}, d)$ value is the best. In that way, the node x makes a locally greedy decision by forwarding the packet to the neighbor so that the Q-value is the best one.

4.3 Our proposal: the QoS forwarding strategy

The goal of routing in IP networks is to calculate the shortest path between each pair (Source, Destination), while considering the cost of each link, in a limited time and space. In a similar way, the routing goal in NDN can be defined as calculating the shortest path between each data-consumer and data-sources that have copies of the data requested by the consumer, by associating a cost to each link in a limited time and space.

In this work, we address the issue of routing with integrated QoS in NDN. Our main aim is to perform routing with quality of service while taking into account the state of the network in a real-time manner. In the sequel, we introduce the functional model used then we present a new forwarding strategy basing on this model.

4.3.1 Functional Model

The NDN network can be considered as a reinforcement learning environment, where each node is a reinforcement learning agent trying to improve the routing by optimizing one or multiple criteria

through the reinforcement function which is unified by the proposed model described below. Consequently, a global function is determined by using multiple parametric functions. These latter are built by a separate module that ensures an action of a local routing decision executed by each agent at node level. In order to reach this goal, each NDN node does continuous reinforcement learning of the network parameters. The rule used for updating the parameters of the network model is:

$$w(n) = w(n-1) - \epsilon(n)F(x(n), w(n-1) \tag{4.3.21}$$

where $w(n)$ is the global objective function, n is the time, $x(n)$ is the current network situation and ϵ is the modification step. F is either the gradient of the global objective function or a heuristic rule. x can be defined by a probability density function $p(x)$, and w the parameters of the learning system. The concept behind our system is based on the fact that each iteration uses an instance of a real flow instead of a finite training set. The goal of this learning system is to find the minimum of a function $C(w)$ called the expected risk function [94] and the average update is therefore a gradient algorithm which directly optimizes $C(w)$. For a given state of the system, we can define a local cost function $J(x, w)$ which measures how well our system behaves on x. The goal of learning is often the optimization of some function of the parameters and of the concept to learn which we will call the global cost function. It is usually the expectation of the local cost function over the space X of the concept to learn:

$$C(w) = \int_x J(x, w)p(x)dx = E_X J(x, w) \tag{4.3.22}$$

Most often we do not know the explicit form of $p(x)$ and therefore $C(w)$ is unknown. Our knowledge of the random process comes from a series of observations x_i of the variable x. We are thus only able to calculate $J(x, w)$ from the observations x_i. A necessary condition of optimality for the parameters of the system is:

$$\triangle C(w) = E_X \triangle_w J(x, w) = 0 \tag{4.3.23}$$

where $\triangle$ is the gradient operator. Since we do not know $\triangle C(w)$ but only the result of $\triangle_w J(x, w)$, we cannot use classical optimization techniques to reach the optimum. One solution is to use an adaptive algorithm as follows:

$$w(n) = w(n-1) - \gamma(n) \triangle_w J(x(n), w(n-1)) \tag{4.3.24}$$

where γ is the gradient step. The (4.3.21) and (4.3.24) are very similar because learning aims to minimize an objective function. The convergence proof of the online gradient algorithm from this formula can be found in [95] where it is proven that this formulation is particularly adequate for describing adaptive algorithms that simultaneously process an observation and learn to perform better. Such adaptive algorithms are very useful in tracking a phenomenon that evolves over time.

4.3.2 Designing the QoS forwarding strategy

We consider three QoS parameters: cost, bandwidth (denoted BW), and Round Trip Time (denoted RTT). Therefore, the forwarding decisions for Interests and Data packets will be performed according to a function $f(cost, BW, RTT)$. It is worth pointing out that the first metric is static, whereas the two last metrics are dynamic. In the sequel, we detail how these parameters are estimated, and how they are used to make forwarding decisions.

4.3.2.1 Criterion 1: Bandwidth

In order to satisfy the required bandwidth of the user or application, when a router receive an Interest, our forwarding strategy allocates the necessary bandwidth resource from outgoing interfaces, before sending the incoming Interest to the next hop(s) on upstream. This process will be performed by all routers traversed by the incoming Interest for all the links of the path connecting the consumer to the data-source. Our forwarding strategy manages the available links bandwidth from its both interfaces. Thus, the capacity C of the link that connects router X using i with router Y via j, is managed from the interfaces i and j. i.e., it reserves and releases the required bandwidth from both i and j.

$$BW(X)_{intiTotal} = BW(Y)_{int-j-Total} = C \tag{4.3.25}$$

$$BW(X)_{int-i-Total} = BW(Y)_{int-j-Total} = C \tag{4.3.26}$$

where $BW(X)_{int-i-Total}$ is the bandwidth capacity of the interface i of the router X.

$$BW(X)_{int-i-res} = BW(Y)_{int-j-res} = BW_{link_{res}} \tag{4.3.27}$$

where $BW(X)_{int-i-res}$ is the residual bandwidth of the interface i of the router X, and $BW_{link_{res}}$ is the residual bandwidth of the link that connect X via i to Y with interface j.

1.1) Residual bandwidth of links

In the simplified scenario where Interests flow in only one direction and data return in the opposite direction, let consider the link that connect the node X via the interface i to node Y with interface j. The residual bandwidth of this link can be calculated, when n Interests are sent from X to Y via the interface i in a given time t as follows:

$$BW(X)_{int-i-res} = BW(X)_{int-i-Total} - \sum_{k=1}^{n} BW_{req_k} \tag{4.3.28}$$

where BW_{Req_k} is the required bandwidth for the Interest k which must be reserved.

However, in NDN the Interest and Data packets flow in both directions simultaneously, because the difference between servers and clients is obscure and every network entity can be a potential content source [66]. In order to ensure that the returned data meet the required QoS and to avoid congestion, our forwarding strategy must take into account the bandwidth consumed in both directions before sending any Interest to upstream. Let assume that at a given time t there are n Interests have been sent from node X to node Y via interface i and there are m Interests that have been sent from the node Y to the node X via interface j. The residual bandwidth of the link between X and Y can be calculated as follows:

$$BW(X)_{int-i-res} = BW(Y)_{int-j-res} = C - (\sum_{k=1}^{n} BW_{req_k} + \sum_{k=1}^{m} BW_{req_k}) \tag{4.3.29}$$

In order to have the real residual bandwidth of the link in its both interfaces, the reservation of the required bandwidth is performed from outgoing interface i of router X before sending the Interest on upstream. Furthermore, the updating of the interface j of router Y is performed after receiving the Interest. However, the forwarding strategy decisions cannot be based only on the residual bandwidth of the link connecting the current node to the next hop, because this latter cannot ensure the retrieving of the data from the next hop if the required bandwidth cannot be satisfied by all interfaces of the upstream router. Consequently, our forwarding strategy uses the estimated residual bandwidth for the entire path from the current node to the node that has a copy of the requested Data.

1.2) Estimated residual bandwidth of a path

The choice of the output interface for the incoming Interest is done by the forwarding strategy on the basis of several QoS criteria. The residual bandwidth is not the only criterion to consider, but the required bandwidth must be satisfied. We used a reinforcement learning approach to compute the path estimated residual bandwidth for the interfaces and for each data name prefix FIB table, which connect the current router to a node that has a copy of the requested Data. Let assume that an Interest is sent from node X to the next upstream node Y via interface k selected by the forwarding strategy. When the node X receive the requested Data packet, it receives also with it, a reinforcement signal of the path estimated bandwidth traversed during the round-trip, by the Interest in round and by Data in trip, from the node Y to the node where the requested Data was retrieved.

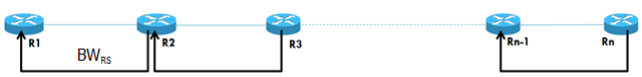


Figure 4.3: Updating the residual bandwidth of a path.

As shown in Fig. 4.3, the estimated residual bandwidth along the path $R_1, R_2, R_3, ..., R_n$ is calculated as follows:

$$BW(R_i)_{int-k-est} = \min(BW(R_i)_{int-k-res}, BW_{RS}) \tag{4.3.30}$$

where T is the reinforcing signal received by R_i with the Data packet, and is defined as follows:

$$BW_{RS} = \min(BW(R_{i+1})_{int-x-res}, BW(R_{i+2})_{int-k-est}) \tag{4.3.31}$$

1.3) Updating the estimated bandwidth

In our forwarding strategy, the bandwidth updating process is triggered by the following three events: receiving an Interest, receiving a Data packet, or receiving a *Nack* Interest. In the sequel, we detail each case.

1.3.a) After receiving an Interest: The reception of an Interest triggers the process of optimal interface selection in order to retrieve the data with the best QoS. Once the forwarding strategy selects the optimal interface, it reserves the required bandwidth from the selected interface, and updates the bandwidth of the incoming interface of the Interest. Furthermore, it sends the Interest to the upstream node. If k is the incoming interface of the Interest and s the selected optimal interface, the updating and the reservation are performed as follows:

$$BW(X)_{int-k-res} = BW(X)_{int-k-res} - BW_{req} \tag{4.3.32}$$

$$BW(X)_{int-s-res} = BW(X)_{int-s-res} - BW_{req} \tag{4.3.33}$$

$$BW(X)_{int-s-est} = \min(BW(X)_{int-s-est}, BW(X)_{int-s-res}) \tag{4.3.34}$$

1.3.b) After receiving a Data After receiving a Data packet, the forwarding strategy releases the required bandwidth from the incoming interface of the Data packet and updates the path estimated bandwidth connecting the current node to the node where the Data packet come from. Furthermore, the forwarding strategy releases the required bandwidth reserved from all requesting interfaces, before sending the Data packet to downstream. The following equations detail how the forwarding strategy releases the reserved bandwidth from the incoming interface of data k and updates the path estimated bandwidth:

$$BW(X)_{int-k-res} = BW(X)_{int-k-res} + BW_{req} \tag{4.3.35}$$

$$BW(X)_{int-k-est} = \min(BW(X)_{int-k-res}, BW_{RS}) \tag{4.3.36}$$

The reserved bandwidth from each interface k in Requesting Faces list is released as follows:

$$BW(X)_{int-k-res} = BW(X)_{int-k-res} + BW_{req} \tag{4.3.37}$$

Generally, we can define the reinforcement signal that the current node will receive to update the estimated bandwidth, as the minimum between the estimated bandwidth of the link that connects the next upstream node to the following node, and the bandwidth estimated of the entire path that connects it to the one node that has the requested Data packet.

1.3.c) After receiving an Interest *Nack* After receiving an Interest *Nack* at each node, the forwarding strategy releases the required bandwidth from the incoming interface of the Interest *Nack* and updates the path estimated bandwidth connecting the current node to the node that has a copy of the requested data as follows:

$$BW_{int-k-res} = BW_{int-k-res} + BW_{req} \tag{4.3.38}$$

$$BW_{int-k-est} = \min(BW_{int-k-res}, BW_{RS}) \tag{4.3.39}$$

If there is an interface that can satisfy the QoS requirements, the forwarding strategy reserves the required bandwidth from the new selected interface as in Section 4.3.2.1. Otherwise, the forwarding strategy releases the reserved bandwidth from all requesting interfaces as in Section 4.3.2.1.

4.3.2.2 Criterion 2: Round Trip Time (RTT)

For a given prefix in the FIB table of a node, RTT is the time elapsed between the Interest sending and the data reception. In ICN, the retrieval of data is done through the path traversed by the Interest during the search of data, from the consumer to ordinary source of data as provider, repository or caching node, but in the opposite direction. This path is constructed according to the various decisions taken by the forwarding strategy at all traversed nodes. In addition, the data may be also found before the producer as intermediate nodes use in-network caching. Therefore, it is clear that the RTT computed for the same prefix of the same node changes according to the retrieval point (where the data was found).

As the requested data could be found in the temporary cache of intermediate nodes, or permanent sources (repository or provider), the estimated RTT would take different values. A natural question is: which estimated value of RTT should be considered in the forwarding decision: the last value, the largest one, the average, etc? We consider two values: the real RTT, and the estimated RTT (RTTest). We exploit an approach based on learning by reinforcement (Q-learning) to compute these values.

The RTT estimated of a given prefix for a given interface is the estimated time taken from the sending of the Interest via this interface until the reception of data via the same interface, through the path connecting the current node to a node that has a copy of the requested data. Consequently, the RTT it is the sum of the transmissions time, waiting time in waiting queue, and processing time at all nodes traversed by the Interest in round, and the data in trip through the optimal path

between the current node and the node where the data has been found. Suppose $RTT(P, X, Y)$, the RTT that the node X estimates to retrieve the Data packet that matches the prefix P through the neighboring node Y connected to X via the interface k from an ordinary source of data (provider, repository, or cache).

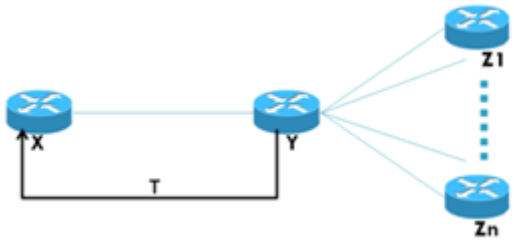


Figure 4.4: Updating the RTT reinforcement signal.

We explain below how to calculate the RTT using the approach detailed in [93]. Let us consider the scheme presented in Figure 4.4.

$$T = \min_{(Z \in neighbors\ of\ Y)} RTT(P, Y, Z) \tag{4.3.40}$$

where T represents the reinforcing signal that the node X will receive upon the reception of data, and $\mu \in [0, 1]$ represents the learning rate. The difference in time between the $oldRTT$ and the $newRTT$ for the same prefix P from node X via node Y is defined as:

$$\Delta RTT(P, X, Y) = \mu(RTT_{New}(P, Y, Z) - RTT_{Old}(P, Y, Z)) \tag{4.3.41}$$

where the new RTT ($RTT_{New}(P, X, Y)$) and the old RTT ($RTT_{Old}(P, X, Y)$) estimated for receiving the Data packet corresponding to the prefix P are related as follows:

$$RTT_{New}(P, Y, Z) = RTT_{Old}(P, Y, Z) + \Delta RTT(P, X, Y) \tag{4.3.42}$$

We replace $\Delta\ RTT(P, X, Y)$ by its value and we have:

$$RTT_{New}(P, Y, Z) = RTT_{Old}(P, Y, Z) + \mu(RTT_{New}(P, Y, Z) - RTT_{Old}(P, Y, Z)) \tag{4.3.43}$$

If we replace $RTT_{New}(P, Y, Z)$ by its value, we find:

$$RTT_{New}(P, Y, Z) = RTT_{Old}(P, Y, Z) + \mu(RTT_{XY} + T - RTT_{Old}(P, Y, Z)) \tag{4.3.44}$$

where RTT_{XY} is the RTT between the two neighboring nodes X and Y. We replace T by its value in equation 4.3.44, we get:

$$RTT_{New}(P,Y,Z) = RTT_{Old}(P,Y,Z) + \mu(RTT_{XY} + \min_{(Z \in neighbors\ of\ Y)} RTT(P,Y,Z) - RTT_{Old}(P,Y,Z)) \tag{4.3.45}$$

Finally, the new RTT is computed as follows:

$$RTT_{New}(P,Y,Z) = RTT_{Old}(P,Y,Z)(1-\mu) + \mu(RTT_{XY} + \min_{(Z \in neighbors\ of\ Y)} RTT(P,Y,Z)) \tag{4.3.46}$$

The updating of the RTT (real RTT or estimated RTT) is triggered by the reception of a Data packet, and it is performed as explained above. In the following subsections, we detail how our forwarding strategy uses the above-mentioned QoS parameters to make forwarding decisions.

4.3.2.3 Exploration and exploitation phases

To implement the proposed strategy, it is necessary to adapt the node data structures in order to meet our forwarding strategy design requirements. Consequently, we must add some fields to the node data structures, which help to save the QoS parameters metrics collected from the network.

Initially, the values of these fields are not available for all data name prefixes in FIB table entries, as well as in the case when a new data name prefix is added to the FIB table. Consequently, the proposed forwarding strategy starts by initializing these fields. The aim of this phase, called the exploration phase, is to evaluate the QoS parameters of each data name prefix in the FIB table for all available interfaces to get the best estimation of the real-time network status, and save them in the node data structures fields dedicated for this aim. To this end, during the exploration phase, our forwarding strategy sends the incoming Interest in round manner via the available interfaces found in FIB entry corresponding to the requested data name. Furthermore, it can retrieve the user requested data, but without ensuring the best QoS. This phase will continue during time duration T (set manually). Once this period is finished, our proposed forwarding strategy switches automatically to the exploitation phase where it starts to use the saved QoS parameters metrics in the process of forwarding making decision. So in this phase, the incoming Interest will forward, via the interface that allows retrieving the requested Data, with the best QoS. Our forwarding strategy updates only the node data structure fields dedicated to save the QoS parameters corresponding to the interface concerned by one of the following events: reception of Interest, Data or Interest Nack. Figure 4.5 shows how the proposed forwarding strategy selects which interface to use to forward the incoming Interest in each phase.

In order to reach the best exploitation of the network and to offer the best QoS for users during the exploitation phase, the proposed strategy forwards the incoming Interest via the interface which

has the smallest RTT among the interfaces, in the data name prefix corresponding to the incoming Interest in the FIB table entries, that can satisfy the required QoS, if the filtered interfaces are updated for times less than DT. Otherwise, the incoming Interest will be forwarded via an available interface that was not updated for more than DT in order to explore and update it. This process is necessary, because it is possible to find an interface that is able to retrieve the requested data with a better QoS than the current best interface. The goal of this process is to update all the interfaces of the data name prefix which was not recently used (more than DT).

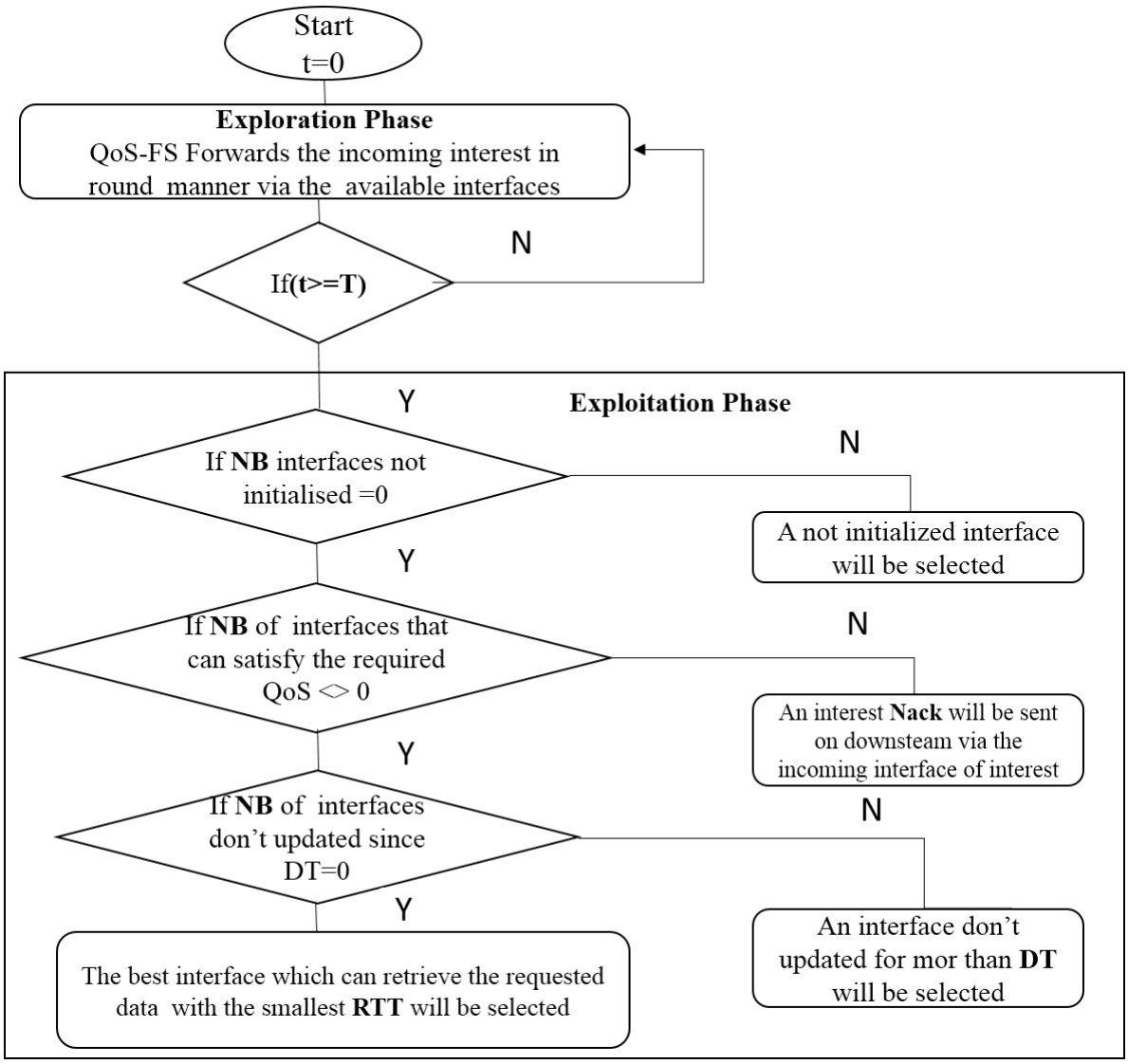


Figure 4.5: Exploration and exploitation phases.

4.3.2.4 Forwarding decision-making algorithms

In this section, we detail how the proposed strategy reacts to the three main events: reception of Interest, Data, or Nack. Furthermore, we present the algorithms that it follows upon reception of Interest, Data and Interest Nack.

4.1) After receiving an Interest: After receiving an Interest, the proposed forwarding strategy performs the following steps: First, it checks the Content Store (CS) for data that match the incoming Interest name. If matched data is found, then it will be sent on downstream via the incoming interface of the Interest and the Interest will be dropped as it is considered satisfied. If no data was found in the CS, the node looks in PIT entries. If an entry is found, the forwarding strategy updates the bandwidth of the incoming interface of the Interest as explained above and adds the incoming interface to the requesting interfaces list, then it drops the Interest. When the requested data is found, a copy will be sent back via incoming interfaces of Interest. If the Interest name does not exist in the PIT, our forwarding strategy creates new PIT entry and looks up for the Interest name in the FIB table. If no FIB entry matches the Interest name, the Interest is deleted. Otherwise, this later will be forwarded as explained in the sequel.

Exploration Phase: in this phase the proposed strategy forwards the incoming Interest in round manner via an available interface found in the data name prefix of the FIB entry corresponding to the requested data name.

Exploitation Phase: in this phase all forwarding decisions of incoming Interests are made according to the required QoS and the interfaces QoS parameters metrics of the data name prefix corresponding to the requested data in the FIB table. The cost parameter is ignored. Thus, we take into account only the two dynamic parameters (RTT and the residual bandwidth). Consequently, each interface has two QoS parameter values, the estimated RTT and the estimated path residual bandwidth connecting the node via the interface to a node that has a copy of the requested data which are saved in the node data structure fields.

The decision-making process is as follows: first, our forwarding strategy filters interfaces that are able to forward the incoming Interest among all available and functional interfaces. So, the interfaces that have an estimated residual bandwidth less than the required bandwidth, or an RTT that exceeds the required RTT will be ignored. Furthermore, if there is no interface that can satisfy the required QoS, then an Interest *Nack* will sent to downstream via the incoming interface of the Interest with the error code *no path*. However, if there are some interfaces that can satisfy the required QoS, the forwarding strategy selects the interface that has the best (i.e., smallest) RTT as the optimal interface and before sending the incoming Interest via this interface, it reserves the required bandwidth from the selected interface and updates the bandwidth of the incoming interface of the Interest as explained above.

In order to ensure the best performance and retrieve the data in the shortest time, the Interest must be forwarded through the interface that has the smallest RTT. The estimated RTT is the estimated time for retrieving Data, which we compute by a Q-learning approach as explained above. Furthermore, for each data name prefix's interface, we have two values for the RTT parameter: estimated RTT and real RTT. When the real RTT is less than the estimated RTT and *not time out*, the RTT is equal to the real RTT. Otherwise, it is equal to the estimated RTT.

As explained in Section 4.3.2.3, the proposed forwarding strategy uses this selection procedure, if all interfaces in the Data name prefix corresponding to the incoming Interest are updated for times less than DT. Otherwise, it selects an interface which was not updated for more than DT. Furthermore, if there is an interface not initialized, which is the case when a new Data name prefix is published, this later will be selected as first priority.

The Algorithm 2 explains in detail how our forwarding strategy works when a router receives a new incoming Interest.

We assume that the worst-case time complexity of the search in CS, PIT and FIB is $O(sizeoftable)$. let consider the following notation:

- n_c: the size of the content store (CS) table
- n_p: the number of entries in the PIT table
- n_b: the number of entries in the FIB table
- n_f: the number of available interfaces

The worst-case time complexity of the **Algorithm 2** is $O(n)$, where $n = \max(n_c, \max(n_p, n_b + n_f))$.

4.2) After receiving a Data packet: After receiving a Data packet, our forwarding strategy starts by updating the estimated RTT and the real RTT fields of the incoming interface of the Data in the FIB entry that match the incoming Data. Then, it releases the reserved bandwidth from the incoming interface of the Data and updates the estimated path bandwidth from this interface to the node where the Data was retrieved as explained above. Then, it looks up if the incoming Data name exists in the PIT entries. If there is a PIT entry matching, the forwarding strategy releases the reserved bandwidth from all the interfaces of the Requesting Faces list as explained above. Afterwards, it sends a copy of the incoming Data with a reinforcement signal for RTT and another for bandwidth to downstream via all the interfaces of the requesting interfaces list, and the forwarding strategy stores the received Data in the content store. Once the Data is forwarded to downstream nodes, it deletes its PIT entry. In the case where there is no entry in the PIT table that matches the incoming

Algorithm 2: After receiving an Interest

```
1  if ((Interest.name)is found in content store) then
2      Forward(Data,bw,cost,k)
3  else
4      if ((Interest.name) is found in PIT ) then
5          add(Requesting Faces list,k) BW_{int-k-res} ← BW_{int-k-res} − BW_{req}
6      else
7          if ((Interest.name) is found in FIB) then
8              Create new PIT entry(Interest.name,nonce,interface,k)
9              intnb ← 0
10             if (t <= T) then
11                 ***The Inintial exploration Phase***
12                 intnb ← intnb + 1 best_{int} ← selectd in round manner
13                 goto Ref
14             else
15                 ***The exploitation Phase***
16                 rtt_{min} ← rttest of int1 best_{int} ← int1
17                 for (eachfacek∈FIB Entry) do
18                     if ((RTTest = 0) then
19                         **a New Data name prefix entry**
20                         intnb ← intnb + 1
21                         best_{int} ← k goto Ref
22                     if (Last update k > DT ) then
23                         intnb ← intnb + 1
24                         best_{int} ← k goto Ref
25                     else
26                         if (BW_{res} <= BW_{req}) then
27                             if (RTT <= Delay_{req}) then
28                                 if ((realRTT < RTTest) and (not time out)) then
29                                     RTT ← realRTT
30                                 else
31                                     RTT ← RTTest
32                                 if (RTT < rtt_{min}) then
33                                     rtt_{min} ← RTT
34                                     best_{int} ← k
35                                 intnb ← intnb + 1
36             Ref : if (intnb > 0) then
37                 S ← best_int
38                 BW_{int-s-res} ← BW_{int-s-res} − BW_{req}
39                 BW_{int-s-est} ← min(BW_{int-s-res}, BW_{int-s-est})
40                 BW_{int-k-res} ← BW_{int-k-res} − BW_{req}
41                 Update PIT entry with S as out interface Send(Interest,S)
42             else
43                 BW_{int-k-res} ← BW_{int-k-res} + BW_{req} send(NACK,k,cod_error=no path)
44         else
45             drop Interest
```

Data name, the Data packet will be dropped. The Algorithm 3 explains in detail how our forwarding strategy works when a router receives a Data packet.

Algorithm 3: After receiving a Data packet

```
1  if ((Data.name) is found in PIT ) then
2      RTT_New(P,Y,Z) ← RTT_Old(P,Y,Z)(1 − μ) + μ(RTT_XY + T) BW_int−s−res ← BW_int−s−res + BW_req
       BW_int−s−est ← min(BW_int−s−res, BW_RS) T ← RTT_New(P,Y,Z)
3      BW_RS ← BW_int−s−est
4      CS.insert(Data)
5      for (each face k ∈ Requesting Faces) do
6          Forwarder(Data, T, BW_RS, k)
7          BW_int−k−res ← BW_int−k−res + BW_req
8      Erase(PIT.entry)
9  else
10     drop Data
```

The worst-case time complexity of the **Algorithm 3** is $O(n)$, where $n = n_p + n_b + n_f$.

4.3) After receiving an Interest *Nack*: After receiving an Interest *Nack*, our forwarding strategy looks up if the Interest *Nack* name exists in the PIT entries. If there is a PIT entry matching, it starts by releasing the reserved bandwidth from the incoming interface of the Interest *Nack* and updates the path estimated bandwidth from this interface to the node that has a copy of the requested Data as explained before. After that, the forwarding strategy filters again the interfaces that are able to retrieve a requested Data among all available and functional interfaces excepting the incoming interface of the Interest *Nack*. The remaining of the procedure is the same as discussed previously. However, if there are some interfaces that can satisfied the required QoS, then the forwarding strategy will update the outgoing field in the PIT entry with a selected interface rather than creating a new PIT entry. Algorithm 4 describes how how our forwarding strategy works when a router receives an Interest Nack.

The worst-case time complexity of the **Algorithm 4** is $O(n)$, where $n = n_p + n_b + n_f$.

4.4 Performance Evaluation

To evaluate the proposed forwarding strategy **QoS-FS**, we have used the ndnSIM 2.0 simulator [71]. We compare the performance of the proposed **QoS-FS** with the Best Route forwarding strategy [70] that uses the minor number of hops as metric to select the best path to reach the permanent content copies. Our analysis is based on two metrics: *the data delivery time* which represents the time elapsed between Interest sending and data reception, and the $hopscount$ which represents the number of hops traversed by the data to arrive to the consumer. The $Abilene$ topology is used as a core of the simulated network, which is composed of 12 routers. In the simulation scenario, each core router is attached to a variable number between 2 and 4 of consumers

Algorithm 4: After receiving an Interest Nack

```
1  if ((InterestNack.name) is found in PIT ) then
2      BW_{int-s-res} ← BW_{int-s-res} + BW_{int-s-req} BW_{int-s-est} ← min(BW_{int-s-res}, BW_{RS})
       BW_{RS} ← BW_{int-s-est}
3      intnb ← 0
4      if (t <= T) then
5          ***The Inintial exploration Phase***
6          intnb ← intnb + 1 best_{int} ← selectd in round manner
7          goto Ref
8      else
9          ***The exploitation Phase***
10         rtt_{min} ← rtt_{est} of int1
11         best_{int} ← int1
12         for (eachfacek∈FIB Entrys) do
13             if ((RTTest = 0)and(realRTT = 0)) then
14                 ***a New Data name prefix entry***
15                 intnb ← intnb + 1
16                 best_{int} ← k
17                 goto Ref
18             if (Last update k > DT ) then
19                 intnb ← intnb + 1
20                 best_{int} ← k
21                 goto Ref
22             else
23                 if (BW_{res} <= BW_{req}) then
24                     if (RTT <= Delay_{req}) then
25                         if ((realRTT < RTTest) and (not time out)) then
26                             RTT ← real RTT
27                         else
28                             RTT ← RTTest
29                         if (RTT < rtt_{min}) then
30                             rtt_{min} ← RTT
31                             best_{int} ← k
32                         intnb ← intnb + 1
33     if (intnb > 0) then
34         S ← best − int BW_{int-s-res} ← BW_{int-s-res} − BW_{int-s-req}
35         BW_{int-s-est} ← min(BW_{int-s-res}, BW_{int-s-est})  Update PIT entry with S as out interface
36         Send(Interest,S)
37         Drop Nack
38     else
39         for (eachfacek∈RequestingFaces) do
40             BW_{int-k-res} ← BW_{int-k-res} + BW_{req} Forward(Nack,T2,k,cod_error=no path)
41         Erase(PIT. entry)
42 else
43     Drop Nack
```

and 1 repository to store permanent contents. Thus, obtaining a total number of nodes equal to 64. We use a catalog of 100 contents, each content can have between 1 and 4 randomly replicas in the different repositories. The consumers request content according to the law of Zipf (see $ns3 :: ndn :: ConsumerZipfMandelbort$ [34]) for further simulation details, we have the default values for the parameters $q = 0.7$ and $s = 0.7$ which represent respectively the parameter of improve rank and parameter of power.

To evaluate the performance of each strategy, we have carried out several simulations for each of the following consumer request frequency: 10, 20, 30, 50, 70 and 100 Interests per seconds. Then, we compute the mean of the data delivery time and the mean of the hop count for each simulation. After that, we compute the mean of the two metrics for all the simulations and for the same consumer request frequency. Figures 4.6 and 4.7 summarize the mean data delivery time and the mean hops count results of the simulations, respectively.

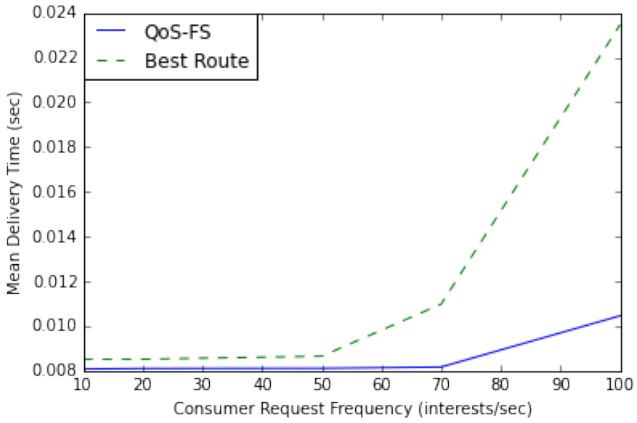


Figure 4.6: Mean delivery time.

Figure 4.6 shows that when the consumer request frequency is lower than 50 for the Best Route, the mean delivery time is almost the same, and between 50 and 70 it increases linearly but beyond that it increases rapidly. For **QoS-FS**, the mean delivery time is almost the same until 70, but after that it starts increases linearly. The obtained results clearly show that the proposed **QoS-FS** can efficiently handle the consumer request frequency as compared to Best Route. Indeed, **QoS-FS** gives always a better delivery time than Best Route.

Figure 4.7 shows the comparison of our proposed **QoS-FS** and Best Route in terms of mean hops count. For the Best Route, it is stable when the consumer request frequency is inferior than 50, and it starts increasing linearly between 50 and 70, but beyond that it increases rapidly exceeding the **QoS-FS** mean hops count which is still always stable. This can be explained by the fact that **QoS-FS** wisely distributes the traffic over all links according to the network state.

Figures 4.6 and 4.7 show that although the mean hops count of Best Route could be inferior

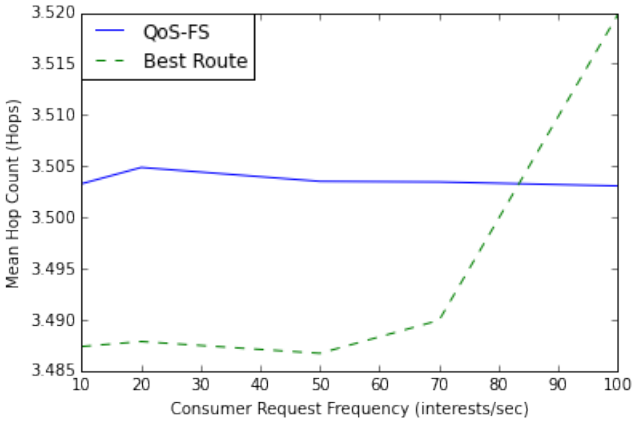


Figure 4.7: Mean hop count.

to that of **QoS-FS**, this latter gives a better mean data delivery time. Thus, the best path is not necessarily the one that has the least hops count.

The obtained results explain the positive effect of the integration of the links real time bandwidth in decisions, which is behind the good distribution of the flow over the available paths. This enables to offer the good delivery time with a constant hops count, in all the experimentation scenarios.

4.5 Conclusion

In this chapter, the issue of routing with QoS in NDN has been considered. A simple and adaptive forwarding strategy, **QoS-FS**, that ensures QoS-based routing in NDN has been devised. The main idea of our proposal is to monitor and estimate, in real-time, the QoS parameters of interfaces, and integrate them in the decision-making process of forwarding, which is adaptive to the network conditions and preferences of user. Two QoS parameters have been considered, namely: bandwidth and RTT. The simulations results showed that the proposed **QoS-FS** offers to the users a better QoS in term of delivery time than the Best Route and it offers also the smallest hops count when a traffic increases which explains the importance of taking into account the network state in terms of QoS parameters. In the next chapter, we will investigate other QoS parameters by using a meta-heuristic optimization algorithm: Ant Colony.

Chapter 5

Ant Colony based QoS aware Forwarding Strategy for NDN

Contents

5.1 Introduction 93
5.2 Ant colony optimization 94
5.2.1 Meta-heuristic optimization 94
5.2.2 Ant colony algorithms principles 94
5.2.3 Ant system 96
5.2.4 Ant Colony System (ACS) 97
5.3 Our Proposal: the Ant Colony Based QoS-aware Forwarding Strategy 98
5.3.1 Designing AC-QoS-FS 98
5.3.2 Forwarding decision-making algorithms 104
5.4 Performance Evaluation 107
5.5 Conclusion 108

5.1 Introduction

Nowadays, the Internet is experiencing a shift from host-centric communications paradigm to a data-centric communications one. In the former case, the location of a content was more significant than the content itself, which the latter case intends to change. Among the earlier attempts to perform such modification, notably the Peer-to-Peer (P2P) networks and Content Delivery Networks (CDN) shall be mentioned, in which the necessary change for the shift of paradigm was performed at the application layer.

Many studies show that P2P and CDN can enhance QoS (Quality of Service) and QoE (Quality of Experience) of end users. However, they incur high costs and may lead to inefficient solutions. Therefore, more recent attempts such as NDN architecture focus on network layer approaches [2].

How to support smart forwarding of Interests over multipaths while considering QoS parameters is a challenging issue in NDN. To address this challenge, this chapter proposes a new QoS-aware forwarding strategy for NDN. Borrowing techniques from the ant colony optimization, the proposed strategy, which is called Ant colony based QoS-aware forwarding strategy (**AC-QoS-FS**), makes full use of both forward and backward ants to rank interfaces. Forward and backward ants (Interest and Data packets) probe real-time network QoS parameters to update the ranking of interfaces in order to select the best one for forwarding the incoming Interests. The effectiveness of **AC-QoS-FS** is validated through ndnSIM simulator. Since meta-heuristics do not require the knowledge of an analytical functional model of the objective function to be optimized and they treat the problem as a black box, the presentation of the analytic model of the network in this chapter does not take place. The rest of this chapter is organized as follows. Ant colony optimization is presented in section 5.2. The proposed QoS-aware forwarding strategy **AC-QoS-FS** is described in Section 5.3 and its performances are evaluated in Section 5.4. Finally, Section 5.5 concludes the chapter.

5.2 Ant colony optimization

The ant colony algorithms are meta-heuristics optimization techniques inspired from collective behavior of real ants in their process of deposition and tracking paths.

5.2.1 Meta-heuristic optimization

Meta-heuristics are high-level procedures designed to solve difficult optimization problems. These are usually problems with incomplete, uncertain, noisy data or faced to limit computing capacity. Meta-heuristics are used in different fields with success. This is due to the fact that they can be applied to any problem that can be expressed in the form of an criterion optimization problem. These methods are inspired from physics (simulated annealing), biology (evolutionary algorithms) or ethology (particle swarm, ant colony).

5.2.2 Ant colony algorithms principles

These algorithms were proposed for the first time by Dorigo *et al.* [96], [74], based on the work of Deneubourg et al. [97], who modeled the random behavior of ants. The first optimization algorithm based on ant colonies, entitled ”Ant System” (AS), was proposed in [74] to solve the Problem of the Traveling Salesman. This latter consists on determining the shortest path that allows a salesman to visit several cities one time each, knowing that the distances between cities are not identical, and

that he ends his journey at the departure city. Since then several improvements and variants to such problem have emerged and have been applied in multiple areas, with different degrees of success. It was pointed out in [98] and [99] that real ants are able to take the shortest path between their nest and a source of food through collaborative collective behavior, knowing that none of the ant has a global view of the path. The behavior of ants is based on the following principles:

- **Self-organization:** the ants are able, by performing simple tasks individually, to solve complex problems. The simple and local behavior of each individual brings out a global pattern through multiple interactions.

- **Stigmergy:** which is a communication technique between individuals (ants) by modifying dynamically the environment in which they operate. Indeed, while moving, ants deposit pheromone to mark the traversed path. In the absence of this substance, ants move randomly into the search space. While in the presence of this substance, the ant detects and follow it according to a proportional probability of its intensity. Thus, the most traversed path becomes most attractive.

- **Decentralized control:** this means that there is no decision taken at one level or by a single individual. Indeed, each individual performs relatively simple actions based only on local information of the environment, without a global view of the problem.

- **A dense heterarchy:** in contrast to a hierarchical structure, where the population is headed by an individual, dense heterarchy is a horizontal structure, where individuals are strongly connected, thus acting on the global properties of the system.

Figure 5.1 illustrates an example of optimization process of the path between an ant nest and a food source. Indeed, at the beginning of the experiment, as seen in Figure 5.1a, the ants arrive at point A, and as there are no traces of pheromone. They will choose one of the two random directions with the same probability. The ants that have taken the **ACD** path will logically arrive faster to the food source than the others and make the opposite route earlier. Thus, the amount of pheromone deposited on the path **ACD** will be greater than that deposited in **ABD**. Since a route containing more pheromone has more chance to be chosen, more ants will choose the **ACD** route (Figure 5.1b). As result, in the long run, the shortest path is reinforced and will be taken by the majority of individuals. In addition, if the phenomenon of pheromone evaporation is considered, after a certain time, all ants will follow the shortest path.

In order to exploit this behavior in an optimization algorithm, an analogy is made between the ant environment and the search space problem, the food source (quality/quantity) and the objective function of the problem, and finally the traces of pheromone and a memory. In addition, some modifications are necessary to have an efficient algorithm. Indeed, an artificial ant (agent) has a

memory, it is not completely blind (has information about the environment) and time is discrete [100].

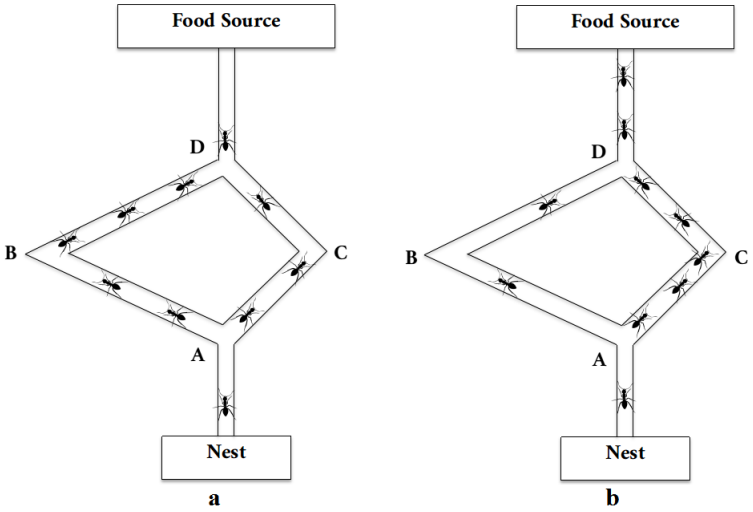


Figure 5.1: Optimization of the path traveled by an ant colony.(a) at the beginning of the search. (b) at the end of the search.

5.2.3 Ant system

In this section, we describe the general principle of the "Ant System" algorithm proposed in [74], applied to solve the Traveling Salesman Problem (TSP). Initially, each ant is placed randomly in one node of a graph(city), then it moves over iterations from one node (city) to another forming a path. At each iteration t, an ant k is chosen to move from the city i where it is located to a not yet visited city j according to the probability $P_{i,j}^k(t)$:

$$P_{i,j}^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}{\sum_{y \in N_i^k} [\tau_{iy}(t)]^\alpha \times [\eta_{iy}]^\beta}, & \text{if } j \in N_i^k \\ 0, & \text{if } j \notin N_i^k \end{cases} \tag{5.2.1}$$

Where:

- $\tau_{i,j}$ is the pheromone density present in the path (i, j) at the iteration t.
- $\eta_{i,j}$ is a visibility measure, which is the inverse of the distance between cities i and j,
- α and β is are parameters which control the influence of the pheromone intensity and the visibility measure, respectively,
- N_i^k is the set of cities neighboring the city i, that are not yet visited by the ant k.

The updating of pheromone intensity is performed after all ants visited all cities(after the end of one cycle) according to the following equation:

$$\tau_{i,j}(t+1) = (1-\rho) \times \tau_{i,j}(t) + \sum_{k=1}^{nb\_ants} \Delta\tau_{i,j}^{k} \tag{5.2.2}$$

where:

- ρ is a pheromone evaporation parameter ($0 \leq \rho \leq 1$),
- nb_ants is the number of ants,
- $\Delta\tau_{i,j}^{k}$ is the amount of deposited pheromone by the ant k over the path (i,j).

  This amount is calculated according to the equation:

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{Q}{L^k}, & \text{if the ant } k \text{ traversed the path } (i,j) \\ 0, & \text{Otherwise} \end{cases} \tag{5.2.3}$$

Where:

- L^k : the length of the path traversed by the ant k,
- Q : a constant of tuning.

5.2.4 Ant Colony System (ACS)

The major contribution in the Ant Colony System algorithm is the introduction of local updating notion [101], [102] in addition to the update that occurs at the end of construction. Indeed, for each ant (individual) at each iteration, an update of pheromone is made at the last visited path, as follows:

$$\tau_{i,j}(t+1) = (1-\varphi) \times \tau_{i,j}(t) + \varphi \times \tau_0 \tag{5.2.4}$$

- φ represents a pheromone evaporation parameter ($0 \leq \varphi \leq 1$),
- τ_0 represents the initial value of the pheromone intensity.

This pheromone evaporation in the traversed path promotes diversification, encouraging ants to visit new paths. The global update of the pheromone is performed at the end of each iteration according to the following equation:

$$\tau_{i,j}(t+1) = \begin{cases} (1-\rho) \times \tau_{i,j}(t) + \rho \times \Delta\tau_{i,j}, & \text{if the ant traversed the path } (i,j) \\ \tau_{i,j}(t), & \text{Otherwise} \end{cases} \tag{5.2.5}$$

In addition, during the construction process, the proportionality pseudo-random rule is used. Indeed, the probability for an ant to move from the city i to the city j is determined by roulette wheel selection.

5.3 Our Proposal: the Ant Colony Based QoS-aware Forwarding Strategy

We point out that the NDN architectures present a symmetric packet routing, i.e., Interest packets and the respective Data packets flow through the same path, but in opposite directions. This behavior matches the natural behavior of ants while searching for the shortest path between their nest and food source. Based on this idea, the ant colony algorithm (the ant system) was originally used to solve the famous NP-complete Traveling Salesman Problem (TSP). Following the same principle, we adapted this algorithm to the design of our proposed forwarding strategy to solve the issues of QoS-aware routing in NDN networks. The main idea of our proposal relies on ants measuring the real-time network QoS parameters of the path they traversed from the data source (producer) up to the data requester (consumer). We use then these measurements to compute the amount of pheromone that will be deposited, in order to select the best interface to forward the incoming Interests. When arriving at each node of the path, an ant deposits a certain amount of pheromone for that correspondent data incoming interface. This amount is determined according to the measured QoS parameters of the path (bandwidth, cost, and RTT), from the data source to the current node. These amounts of pheromone will be used for ranking interfaces in order to select the best one for forwarding the incoming Interests. The ultimate goals that we aspire by the proposed forwarding strategy are: i) to maintain a best data delivery performance, ii) to get a best traffic distribution over the network links, and iii) to avoid network instability. Our proposed Ant Colony based QoS-aware Forwarding Strategy for NDN will be called **AC-QoS-FS**. In the sequel, we will explain in details the **AC-QoS-FS** design and the associated algorithms.

5.3.1 Designing AC-QoS-FS

Let $G(X, U)$ be the graph representing the network topology where $X = (x_1, x_2, ..., x_n)$ ($|X| > 0$) is a set of NDN nodes, and $U = u_1, u_2, ..., u_m$ ($|U| > 0$) is a set of links connecting such nodes. The search space is the set of the available paths in the graph $G(X, U)$ representing the NDN network. There are three kinds of nodes:

- **Consumer (nest)**: it is similar to an ant nest. In our proposal there are several nests, where they express their food needs that match the data by their names.
- **Producer (food source)**: a data storage server able to respond to various arriving demands. It represents the foods source, where ants can get their food needs (data).
- **Router (city)**: It is responsible to direct the Interests packets towards the potential sources of the requested data at first hand, i.e. the Interest ants to the food source, and directs the Data packets toward the appropriate consumer, by exploiting data names, i.e. data ants to the nest. It is a seam connecting cities in TSP. It makes difference between packets.

In our proposal the ant is a packet which is identified by the data name, being either an Interest packet in the search phase of requested data, or a Data packet, after meeting the requested data, upon the transmission data phase toward the consumer(s). Table 5.1 and Figure 5.2 relate the nomenclature in NDN with the Ant Colony and TSP models.

NDN Network	**Ant Colony**
Packet (Interest - Data)	Ant
Consumer	Nest
Producer or caching router	Food source
Routers	Cities
Links	Routes between Cities

Table 5.1: Relation between NDN and Ant Colony.

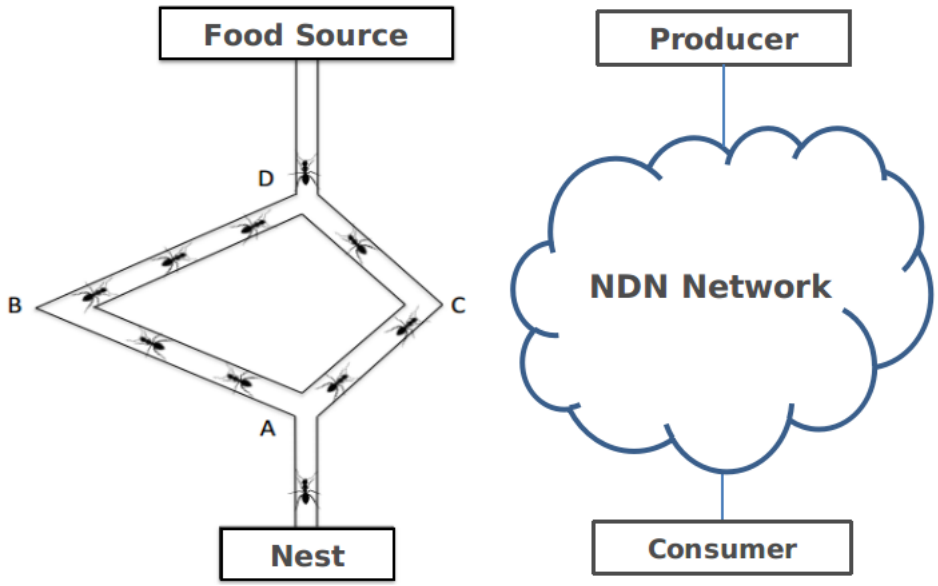


Figure 5.2: Similarities between Ant Colony and NDN network.

Three QoS parameters are considered: cost, bandwidth, and Round Trip Time (RTT). Therefore, according to the values of these parameters, the forwarding decisions for Interests and Data packets will be performed. Real ants have a very limited memory. The virtual ants used for the TSP algorithm, however, can store the cities list that they traversed. In our proposal, the ant has also a memory, which allows it to store the network QoS measurements of the path traversed from the food source toward the nest. In other words, each ant, when returning toward its nest, stores the residual bandwidth and the cost of the path traversed between food source and last node. These information will be used to compute the pheromone amount that will be deposited in the FIB entry corresponding to the data handled by the ant (in the field corresponding to the interface of the

incoming data). In this section, we first detail how these parameters are estimated, then how the proposed forwarding strategy used them in the forwarding decisions.

5.3.1.1 Criterion 1: Bandwidth

The proposed forwarding strategy manages the available links bandwidth from interfaces. Thus, the capacity $C1$ of the link that connects router X with router Y by using the interface i, is managed via the interface i and the capacity $C2$ of the link that connects router Y with router X via j is managed via the interface j.

$$BW(X)_{int-i-Total} = C1$$

$$BW(Y)_{int-j-Total} = C2$$

where $BW(X)_{int-i-Total}$ is the bandwidth capacity of the interface i of the router X and $BW(Y)_{int-j-Total}$ is the bandwidth capacity of the interface j of the router Y. We have also:

$$BW(X)_{int-i-res} = BW_{link_{res}(X,Y)}$$

$$BW(Y)_{int-j-res} = BW_{link_{res}(Y,X)}$$

where $BW(X)_{int-i-res}$ is the residual bandwidth of the interface i of the router X that connects X to Y, and $BW_{link_{res}(X,Y)}$ is the residual bandwidth of the link that connects X via i to Y, where $BW(Y)_{int-j-res}$ is the residual bandwidth of the interface j of the router Y that connects Y to X, and $BW_{link_{res}(Y,X)}$ is the residual bandwidth of the link that connects Y to X via interface j.

The proposed forwarding strategy allocates the necessary bandwidth resource from outgoing interfaces, before sending the incoming Interest to the next hop on upstream. This process will be performed by all routers traversed by the incoming Interest for all links of the path connecting the consumer to the data-source.

1.1) Residual bandwidth of links:

Let us consider the link that connects the node X to node Y via the interface i. The residual bandwidth of this link is equal to the residual bandwidth of the interface i. It can be calculated, when n Interests are sent from X to Y via the interface i in a given time t as follow:

$$BW(X)_{int-i-res} = BW_{link_{res}(X,Y)} = BW(X)_{int-i-Total} - \sum_{k=1}^{n} BW_{req_k} \tag{5.3.6}$$

where BW_{Req_k} is the necessary bandwidth for the Interest k which must be reserved.

1.2) The path residual bandwidth:

Let us assume that an ant (Interest) has been forwarded from node R_i to the next node on upstream R_{i+1} via the interface k selected by the forwarding strategy. While the ant (Data) returns back to the node R_i, it also bring information concerning the quality of the path traversed from the R_{i+1}

to R_i, such as the minimum bandwidth BW_{min} of the current link. This information is used to determine the amount of pheromone that the ant will depose at the interface connecting such link.

From Figure 5.3, the path between a node R_i, $i \in [1, n]$ and the data source R_n and the residual bandwidth of an interface k of the node R_i for a data name prefix P in the FIB table, can be calculated as follows:

Figure 5.3: Updating the residual bandwidth of a path.

$$BW(R_i, P)_{int-k-path} = \min(BW(R_i)_{int-k-res}, BW_{TP}) \tag{5.3.7}$$

where

$$BW_{TP} = \min(BW(R_{i+1})_{int-x-res}, BW(R_{i+2})_{int-k-path}) \tag{5.3.8}$$

1.3) The path residual bandwidth update:

In our forwarding strategy, the bandwidth updating process is triggered by the events: receiving an Interest or receiving a data packet.

1.3.1) After receiving an Interest:

The event of reception of an Interest triggers the selection interface process. Once the interface is selected, the forwarding strategy reserves the necessary bandwidth from this interface. Furthermore, it sends the Interest to the upstream node. The reservation is performed as follows:

$$BW(X)_{int-k-res} = BW(X)_{int-k-res} - BW_{req} \tag{5.3.9}$$

where k is the selected interface and BW_{req} is the necessary bandwidth.

1.3.2) After receiving a data:

The proposed forwarding strategy releases the reserved bandwidth from the incoming interface of the ant and updates the FIB table entry corresponding to the incoming data name (in the field that matches the incoming interface of the ant), the residual bandwidth of the path that connects the current node to the source of the data. The releasing is performed as follows:

$$BW(X)_{int-k-res} = BW(X)_{int-k-res} + BW_{req} \tag{5.3.10}$$

where k is the incoming interface of the ant and BW_{req} is the bandwidth reserved at the search phase. The estimated path residual bandwidth, if BW_{TP} is the minimum bandwidth of the links that the ant traversed from the food-source to the current node, is computed as:

$$BW(X)_{int-k-path} = \min(BW(X)_{int-k-res}, BW_{TP}) \tag{5.3.11}$$

5.3.1.2 Criterion 2: Cost

We assume that the cost of the link connecting the nodes X and Y via the interface i equals the cost of the interface i. The cost of the path that was traversed by the ant is calculated by adding up the cost of links in that path, or it is also the sum of the nodes incoming interfaces costs that form the path. The cost of the path from the source food to the current node X for k in P is computed as follows:

$$costP(X, P, k) = cost_{TP} + cost(X, Y, k) \tag{5.3.12}$$

where k is the incoming interface of the ant, and P is the FIB data name prefix entry that match the incoming data handled by the ant in the node X to the next hop node on upstream Y via k. $cost(X, Y, k)$ is the cost of the link that connects the node X to the node Y via the interface k and T is the cost of the path that connects Y to the source food.

5.3.1.3 Criterion 3: Round Trip Time (RTT)

For a given prefix P in the FIB table of a node X, RTT is the time elapsed between the Interest sending and the data reception via the interface k. In this proposal, this is computed by a timer starting when sending the ant to the next node on upstream and stopping after receiving the ant that handled the requested data from the same interface k.

$$RTT(X, P, k) = Time_{Receiving} - Time_{Forwarding} \tag{5.3.13}$$

To implement the proposed strategy, it is necessary to adapt the node data structures to our forwarding strategy design requirements. Consequently, we add other fields to the node data structures: one field is used to store the pheromone and the other to save the network QoS parameters metrics collected by the ants. The added fields are:

- **Pheromone**: it stores the amount of pheromone,
- **RTT**: it stores the round trip time elapsed between the departure of the ant and its return from the same interface,
- **BW** and **cost** are used, respectively, to store the residual bandwidth and cost of the path traversed by the ant from the food source to the current node.

Figure 5.4 shows the added fields. They are used to compute the pheromone amount that the ant will depose. The details are described in Section 5.3.1.5.

5.3.1.4 The moving rule

An Interest (ant) located in the node x_i can be forwarded toward one among all nodes connects to x_i. The **AC-QoS-FS** uses the pheromone amount $\tau_{i,j}$ of the interface that connected x_i to x_j to

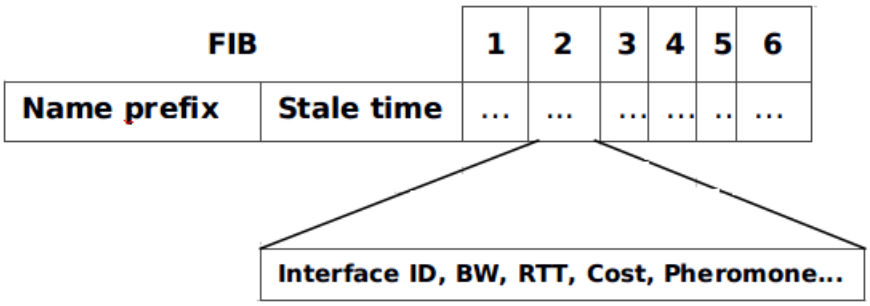


Figure 5.4: Forwarding state in FIB.

compute the probability of the node x_j as next hop. Initially, the pheromone amount is $\tau_{i,j} = \tau_0$. N_{x_i} is a set of nodes connected to x_i except for the node where the incoming Interest comes from ($|N_{x_i}| \geqq 1$), the probability of sending the incoming Interest via the interface j connected to the node x_j is mathematically written as follows:

$$P_{i,j} = \begin{cases} \frac{\tau_{i,j}}{\Sigma_{j \in N_{x_i}} \tau_{i,j}}, & \text{if } j \in N_{x_i} \\ 0, & \text{if } j \notin N_{x_i} \end{cases} \tag{5.3.14}$$

where the transition probabilities $P_{i,j}$ of the node x_i fulfill the constraint $\sum_{j \in N_{x_i}} P_{i,j} = 1$, $i \in [1, |X|]$.

5.3.1.5 Pheromone update

The real ant deposits the pheromone on all the path when it returns back to its nest. In ant colony based TSP algorithm, the ants update the pheromone amount when all the ants finished a cycle (when all the cities had been visited by the ants).

In our proposal, when the ant finds (meets) the requested data, it goes back to the consumer node (starting point or nest), taking the same path traversed during the research phase in opposite direction. Consequently, it crosses the same nodes but in reverse order. Therefore, at each node, it updates the amount of pheromone according to the quality of the path traversed from the food source, which can be a producer or a caching router (content store), to the current node, only in the field corresponding to the incoming interface of the ant in the FIB entry matching the data name handled by the ant. The quality of the path depends on the path QoS parameters measurements that the ant has collected.

Consider that a data (ant) arriving at node x_i through interface k matches the prefix P in the FIB table. The amount of pheromone that the ant will deposit (update) on the correspondent field is computed as follows:

$$\tau_{x_i}(P,k)_(new) = (1-\rho)\tau_{x_i}(P,k)_(old) + \rho\Delta\tau_{x_i}(P,k) \tag{5.3.15}$$

Where $\tau_{x_i}(P,k)$ is the pheromone amount of the interface k for the prefix P in the node x_i. $\Delta\tau_{x_i}(P,k)$ is the amount of pheromone that the ant will deposit on the interface k of the node x_i for the prefix P.

$$\Delta\tau_{x_i}(P,k) = \frac{Q}{c1 * RTT + \frac{c2}{BW} + c3 * cost} \tag{5.3.16}$$

with $c1 + c2 + c3 = 1$. $c1, c2$ and $c3$ are the weighting constants that represent respectively the importance of the RTT, bandwidth and cost. Q is a tuning parameter that will be discussed in Section 5.4. The amount of pheromone that the ant will deposit cannot be computed by equation (5.3.16) because it groups three different type of units and two kind of parameters: BW, which is beneficial criterion whereas the others parameters RTT and cost are expense criteria. So, the values of parameters need to be normalized.

The expense criterion are normalized as: $r_{i,j} = \frac{V_{min}}{V_{i,j}}$. V_{min} is the smallest value of the criterion j for all the interfaces and $V_{i,j}$ is the values of the criterion j for the interface i. Therefore, RTT and cost are normalized respectively as: $\frac{BRTT}{RTT}$ and $\frac{Bcost}{cost}$. $BRTT$ is the best (minimum) RTT of all the interfaces and RTT is the RTT of the interface. $Bcost$ is the best (minimum) $cost$ of all the interfaces and $cost$ is the $cost$ of the incoming interface of ant.

The beneficial criteria are normalized as: $r_{i,j} = \frac{V_{i,j}}{V_{Max}}$. V_{Max} is the biggest value of the criterion j for all the interfaces and $V_{i,j}$ is the values of the criterion j for the interface i. Furthermore, bandwidth is normalized as follow: $\frac{BW}{BBW}$. BBW is the best (Maximum) bandwidth of all the interfaces and BW is the interface bandwidth.

Equation (5.3.16) will be redefined as follows:

$$\Delta\tau_{x_i}(P,K) = \frac{Q}{c1 * \frac{BRTT}{RTT} + c2 * \frac{BW}{BBW} + c3 * \frac{Bcost}{cost}} \tag{5.3.17}$$

5.3.2 Forwarding decision-making algorithms

In this section, we explain in detail how **AC-QoS-FS** reacts to the main events: reception of Interest and data. Furthermore, we present the algorithms that follow upon reception of those packets.

5.3.2.1 After receiving an Interest

The proposed forwarding strategy performs the following steps. First, it checks the Content Store (CS) for data matching the incoming Interest name. If matched data is found then it will be sent on downstream via the incoming interface of the Interest and the Interest will be dropped as it is considered satisfied. If no data were found in the CS, the node looks in PIT entries. If an entry is

found, the forwarding strategy adds the incoming interface to the requesting interfaces list, then it drops the Interest. When the requested data is found, a copy will be sent back via incoming interfaces of Interest. If the Interest name does not exists in the PIT, our forwarding strategy creates a new PIT entry and looks up for the Interest name in the FIB table. If no FIB entry matches the Interest name, the Interest is deleted. Otherwise, there is a FIB entry that matches the requested data name, **AC-QoS-FS** will forward the incoming Interest as explained in 5.3.1.4.

In order to get the best data delivery performance and avoid network instability, the proposed forwarding strategy does not forward the incoming Interest via the interface that has the biggest probability, instead it uses the probabilities computed for all interfaces as inputs for *the roulette wheel selection algorithm* which makes an oriented randomly selection corresponding to the interface probability. Consequently, the proposed forwarding strategy does not always send the incoming Interests via the interface that has the biggest probability (the biggest pheromone amount); however, it gives opportunity to other interfaces that have a small probability (pheromone), which ensures better traffic distribution over the network links.

Once the selection process is finished and the forwarding interface is selected, the proposed forwarding strategy starts by reserving the necessary bandwidth from the selected interface. Then it updates the PIT entry with the selected interface as outgoing interface and finally it sends the Interest. Algorithm 5 explains in detail through a pseudo-code how our forwarding strategy works when a router receives a new incoming Interest.

Algorithm 5: After receiving an Interest

```
1  if ((Interest.name) is found in CS) then
2      Forward(ant, k)
3  else
4      if ((Interest.name) is found in PIT) then
5          add(Requesting Faces list, k)
6      else
7          if ((Interest.name) is found in FIB) then
8              Create new PIT entry (Interest.name, nonce, interface, k)
9              *** Selection process ***
10             for (each face K ∈ FIB Entry) do
11                 Compute the probability of interface K according to Equation 5.3.14
12             Use the roulette wheel to select the outgoing interface S
13             *** Forwarding process ***
14             BW_{int-S-res} ← BW_{int-S-res} − BW_{req}
15             Update PIT entry with S as out interface
16             Send(Interest, S)
17         else
18             Drop Interest
```

We assume that the worst-case time complexity of the search in CS, PIT and FIB is $O(sizeoftable)$. let consider the following notation:

- n_c: the size of the CS table
- n_p: the number of entries in the PIT table
- n_b: the number of entries in the FIB table
- n_f: the number of available interfaces

The worst-case time complexity of the **Algorithm 5** is $O(n)$, where $n = \max(n_c, \max(n_p, n_b + n_f))$.

5.3.2.2 After receiving a Data packet

The proposed forwarding strategy looks up if the incoming data name exists in the PIT entries. If no matching entry is found, the data packet will be dropped. Whenever a PIT entry matching the data handled by the ant is found, **AC-QoS-FS** starts by releasing the reserved bandwidth from the incoming interface of the ant, then it computes the RTT and updates the residual bandwidth and cost of the path between the current node and the node from where the data was retrieved, as explained in Section 5.3. Furthermore, it saves these QoS parameters metrics in the fields corresponding to the incoming interface of the data in the FIB entry matching the incoming data. After that, **AC-QoS-FS** updates the QoS parameters metrics handled by the ant about the path traversed, i.e., the path cost and path residual bandwidth, which are the same values that were saved previously in the fields corresponding to the incoming interface of the ant in the FIB entry matching the data. Next, it computes the pheromone amount and deposits (saves the values) it in the field dedicated for this goal as explained in Section 5.3.1.5. Finally, **AC-QoS-FS** sends a copy of the incoming data (ant) via all the interfaces of the requesting interfaces list to the nodes on downstream, and stores the received data in CS. Once the data is forwarded to downstream nodes, it deletes its PIT entry. Algorithm 6 details how our forwarding strategy works when a router receives a data packet.

Algorithm 6: After receiving a data packet

```
1  if ((Data.name) is found in PIT) then
2      RTT(X,P,S) = Time_Receiving − Time_Forwarding
3      costP(X,P,S) = Ant.cost() + cost(X,Y,S)
4      BW(X)_int−S−res = BW(X)_int−S−res + BW_req
5      BW(X,P,S)_int−S−path = min(BW(X)_int−S−res, Ant.BW())
6      Ant.BW(BW(X)_int−S−path)
7      Ant.cost(cost)
8      Update the pheromone using Equations 5.3.15 and 5.3.17
9      CS.insert(Data)
10     for (each face k ∈ Requesting Faces) do
11         Forward(Ant,k)
12     Erase(PIT.entry)
13 else
14     Drop data
```

The worst-case time complexity of the **Algorithm 6** is $O(n)$, where $n = n_p + n_b + n_f$.

5.4 Performance Evaluation

To evaluate the proposed forwarding strategy **AC-QoS-FS**, we have used the ndnSIM 2.0 simulator [71]. The performances of **AC-QoS-FS** are compared with the best route forwarding strategy [70]. Our analysis is based on the following metrics. The first is the data delivery time, which represents the time elapsed between the sending of Interest and data reception. The second metric is the cost, which represents the sum of links cost traversed by the data packet from the data source to the consumer. The third metric is the hops count, representing the number of hops traversed by the data between data source and the consumer. As fourth, we have the dropped packets in Kilobyte and finally, we evaluate the mean hit ratio. We have used the Abilene topology as a core of the simulated network, which is composed of 12 routers. In the simulation scenario each core router is attached to 1 repository used to store permanent contents and a variable number of consumers between 2 and 4. Thus, the nodes total number is equal to 64. We have used a catalog of 100000 contents, where each content has 3 randomly replicas in the different repositories and each router can store up to 1000 chunks in its CS. The consumers request content according to the largely adopted Zipf's law, concerning the contents popularity, for the parameter values of $\alpha = 0.8; 1.0; 1.2$ [103] with different frequencies: 10, 20, and 30 Interests per second.

It is worth mentioning that the **AC-QoS-FS** results depicted in Figures 5.5(a) , 5.5(b) , 5.5(c) , 5.5(d) , and 5.5(e) , have been obtained with the following configuration: $c1 = c2 = c2 = 1/3$ which means that RTT, bandwidth and cost have the same importance for the application or the end user and we have tuned $\rho = 0.2$ and $Q = 100$. Figure 5.5(a) compares the mean cost associated with data transfers using our strategy and Best Route. For a frequency of 10 requests per second, our strategy is roughly $8\% - 10\%$ higher than Best Route, for the three values of the Zipf-law α parameter. However, there is an interesting trend showing that, the higher the frequency of requests, the lesser the cost of **AC-QoS-FS**. In the case of a frequency equals to 30, the reduction of the costs remains around 10% for the 3 values of α. This shows that for scenarios with increasing loads (several requests flowing), our strategy tends to cost less.

Figure 5.5(b) shows that the hit ratio for our solution is, in general, less important than the other strategy. As a result, the average number of hops an ant must perform is slightly higher for **AC-QoS-FS**, as seen in Figure 5.5(c) . This phenomenon is explained due to the nature itself of our algorithms, which is based on a random selection of the path to be followed by the ant, according to the amounts of pheromone at each router. This implies that, a path leading to a next hop with low likelihood of hit ratio can be also selected. However, Figure 5.5(d) shows that this behavior has no significant impact on the average round trip time, since it remains roughly identical for both solutions. Also important, the rate of dropped packets is similar, even if the ants perform longer

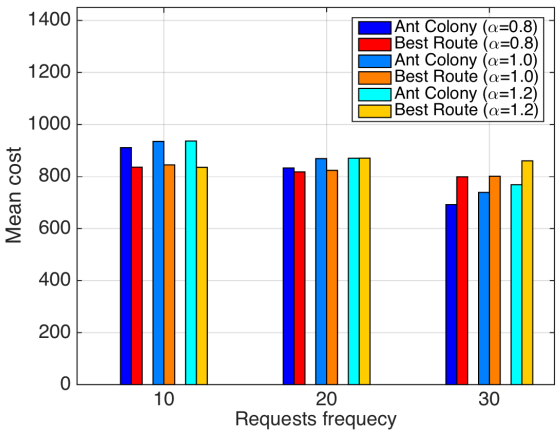
hops to retrieve information, as shown in Figure 5.5(e) .

The results of different scenarios that we had been executed demonstrate that we can design a forwarding strategy that could ensure the best performances by tuning ($c1, c2, c3, \rho$ and Q) parameters according to the application or end user requirements.

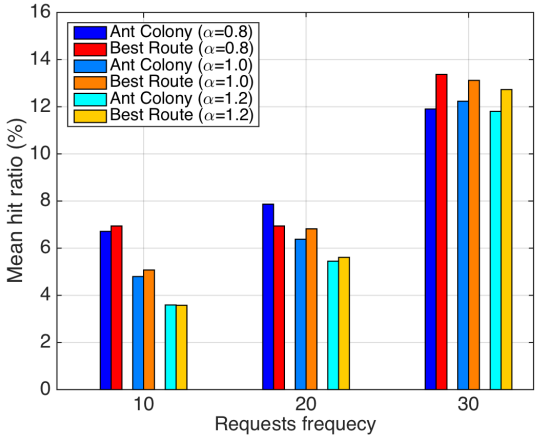
5.5 Conclusion

In this chapter, we presented the **A**nt **C**olony based **Q**uality-of-**S**ervice (QoS)-aware **F**orwarding **S**trategy (**AC-QoS-FS**) designed to solve the issue of routing with QoS in NDN. It is a new NDN adaptive QoS-aware strategy that takes into account the similarities between the natural behavior of real ants and NDN forwarding process. We have considered three QoS parameters: RTT, bandwidth, and cost. The performance evaluation of **AC-QoS-FS** in different simulation scenarios shows the effectiveness of our solution. Furthermore, the obtained results demonstrate that we could design a forwarding strategy that ensure the best performances by tuning ($c1, c2, c3, \rho$ and Q) parameters according to the application or end user requirements.

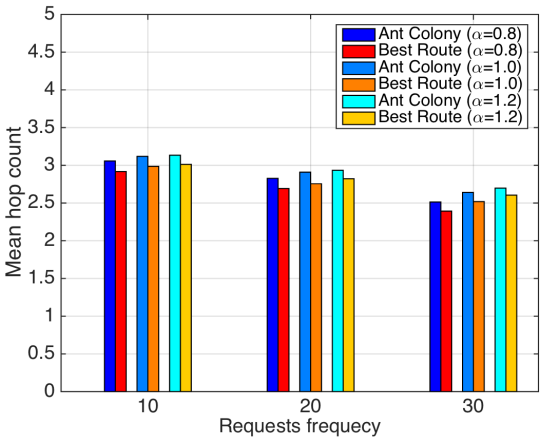
In the next chapter, we will investigate other QoS parameters that are related to congestion issues in NDN networks. So, we will adapt the approach presented in this chapter in order to take into account these new parameters.



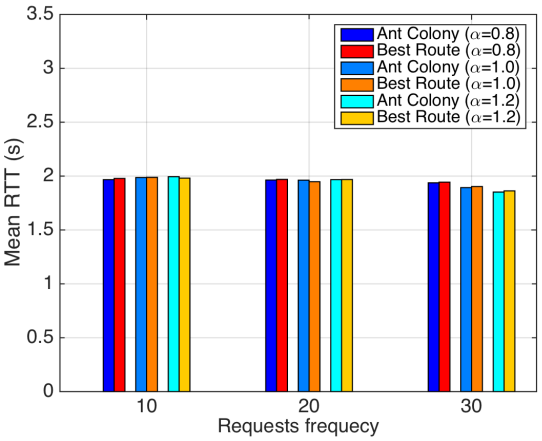
((a)) Mean cost



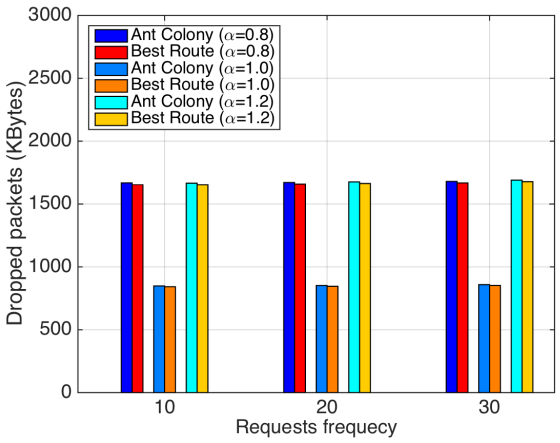
((b)) Hit ratio



((c)) Mean hop count



((d)) Mean delivery time



((e)) Dropped packets in kilobyte

Figure 5.5: Obtained results

Chapter 6

Multi-Criteria QoS-based Forwarding Strategy for NDN

Contents

6.1 Introduction 111
6.2 Our Proposal: Multi Criterion Ant Colony Based QoS-aware Forwarding Strategy(MC-QoS-FS) 112
6.2.1 Designing **MC-QoS-FS** 113
6.2.2 Forwarding decision-making algorithms 117
6.3 Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (FCR-QoS-FS) 119
6.3.1 Designing of **FCR-QoS-FS** 120
6.3.2 Forwarding decision-making algorithms 126
6.4 Performance Evaluation 127
6.5 Conclusion 131

6.1 Introduction

In order to satisfy the requirements of users or applications, it is sometimes preferable to optimize different kinds and number of QoS parameters. In this chapter we start by generalizing the approach proposed in Chapter 5 for supporting n QoS parameters i.e. proposing a model for designing a Multi Criterion Ant Colony QoS based Forwarding Strategy for NDN. After that, by applying the principles proposed in this model, we present the design of a new forwarding strategy called: Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (**FCR-QoS-FS**), which takes into account five QoS parameters. **FCR-QoS-FS** is designed to be adapted as

much as possible to the network dynamic traffic, in particular to the problems of congestion. Consequently, in addition to the QoS parameters taken into account in the Chapter 5 (RTT, bandwidth, and cost), it also takes into account two other parameters IAR and DSR which have a high impact to the congestion. These latter are described respectively in sections 6.3.1.4 and 6.3.1.5. The rest of this chapter is organized as follows: the proposed designing model for Multi Criterion Ant Colony based QoS-Aware Forwarding Strategy **MC-QoS-FS** is described in Section 6.2. The application of this model for designing, named as Flow Congestion Reduction Ant Colony based Quality of Service aware forwarding strategy **FCR-QoS-FS** is presented in Section 6.3. Then **FCR-QoS-FS** performances are evaluated in Section 6.4. Finally, Section 6.5 concludes the chapter.

6.2 Our Proposal: Multi Criterion Ant Colony Based QoS-aware Forwarding Strategy(MC-QoS-FS)

The most important feature of NDN architecture is the inherently support of multipath routing. Furthermore the forwarding of packets in NDN is symmetric, i.e. Interests and Data packets flow across the same path but in opposite directions, through available paths provided by the native multipath forwarding of NDN. This is similar to the natural behavior of ants while searching for the shortest path between their nest and food sources.

Based on this similarity, we adapted the ant colony algorithm (the ant system), that was originally used to solve the famous NP-complete Traveling Salesman Problem (TSP), in order to design our proposed forwarding strategy to solve the issues of QoS-aware routing in NDN networks.

The main idea of our proposal relies on the real-time network QoS parameters measurements collected by the ants about the quality of paths traversed between the data-source (producer or content store) and data requester (consumer). When an ant arrives at each node of the path, it deposits a certain amount of pheromone for the data incoming interface in the FIB data name prefix entry matching the incoming data. This amount of pheromone is determined according the QoS parameters measurements that the ant collected and reflect the quality of the path traversed by the ant from the data-source to the current node. These amounts of pheromone will be used for ranking interfaces, in order to select the best one for forwarding the incoming interests.

In this chapter we will present the designing principles of forwarding strategies that enable the support of n QoS parameters (P_1, P_2,.....,P_n) by adapting the ant colony algorithm in order to offer the operators a real tool. Such tool allows for them to design and implement the appropriate forwarding strategy, that responds to the QoS requirements of users or applications. The ultimate goals that we aspire by the proposed forwarding strategy are: i) maintaining a best data delivery performance, ii) getting a best traffic distribution over the network links and minimizing the amount of dropped packets, iii) avoiding network instability. Our proposed Multi Criteria Ant Colony based QoS-aware Forwarding Strategy for NDN will be called **MC-QoS-FS**. In the sequel,

we will explain in details the **MC-QoS-FS** design and the associated algorithms.

6.2.1 Designing MC-QoS-FS

Let $G(X, U)$ be the graph representing the NDN network topology where X ($|X| > 0$) is a set of NDN nodes, and U ($|U| > 0$) is a set of links connecting such nodes. The search space is the set of the available paths in the graph $G(X, U)$ representing the NDN network. There are three kinds of nodes:

- **Consumer (nest)**: it is similar to an ant nest. In our proposal there are several nests, where they express their food needs that match the data by their names.
- **Producer (food source)**: a data storage server which responds to various arriving demands. It represents the food sources, where ants can get their food needs (data).
- **Router (city)**: it is a seam connecting cities in TSP. It is capable to make difference between the type of packets (Interest and Data). It is responsible to direct the Interests packets towards the potential sources of the requested Data at first hand, i.e. the Interest ants to the food source, and directs the data packets toward the appropriate consumer, by exploiting data names, i.e. data ants to the nest.

In our proposal the ant represents a packet which is identified by the data name, being either an Interest packet in the search phase of requested data, or a Data packet, after meeting the requested data, upon the transmission data phase toward the consumer(s). Table 6.1 and Figure 6.1 relate the nomenclature in NDN with the Ant Colony and TSP models.

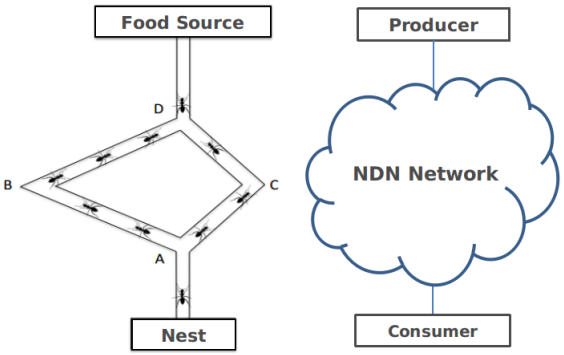


Figure 6.1: Similarities between Ant Colony and NDN network.

By analogy between TSP and forwarding process in NDN network, we can point out the following differences:

- In TSP, the goal consists on finding a Hamiltonian circuit of minimal length on the graph. A Hamiltonian circuit passes exactly once over all the summit of graph, so, in TSP all cities

NDN Network	**Ant Colony**
Packet (Interest - Data)	Ant
Consumer	Nest
Producer or caching router	Food source
Routers	Cities
Links	Routes between Cities

Table 6.1: Analogy between NDN and Ant Colony.

must be visited by ants. While the goal in NDN network consists on retrieving the requested data with the best delivery performance without taking into account the number of nodes visited (no condition about number of nodes must visited).

- In TSP, the salesman traverses all cities forming a circuit, which starts with one city and finishes with the same city. While in NDN network, when the Interest packet finds the requested data, this latter goes back to the requester (departure node) on the same path but in opposite direction.

Let us assume that our forwarding strategy is able to take into account n parameters $P_1, P_2, P_3, ..., P_n$. Consequently, the proposed forwarding strategy makes the forwarding decisions for Interest and data packets according to the values of these parameters. As known, the real ants have a very limited memory. However the virtual ants used in ant colony TSP algorithm can store the cities list that they traversed. In our proposal, the ant has also a memory, that allows it to store the measured network QoS parameters $P_1, P_2, P_3, ..., P_n$ of the path traversed from the food source toward the nest. Consequently, it is necessary to adapt the data packet, in order to be able to handle the network QoS parameters measurements collected $P_1, P_2, P_3, ..., P_n$, by adding other fields respectively to the number of parameters that we want to collect. Figure 6.2 shows the data structure of the Data packet.

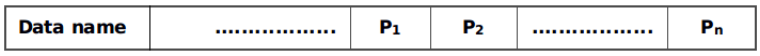


Figure 6.2: The structure of Data packet in **MC-QoS-FS**.

To implement the proposed forwarding strategy **MC-QoS-FS**, it is necessary also to adapt the node data structures to our design requirements. Consequently, we must add other fields to the FIB table entries: one field is used to store the pheromone and other fields to save the network QoS parameters measurements collected by the ants $P_1, P_2, P_3, ..., P_n$. Figure 6.3 shows the new structure and the added fields. The parameters values stored in the data packet (Ant) and in FIB table are used to compute the pheromone amount that the ant will depose as described in Section 6.2.1.2.

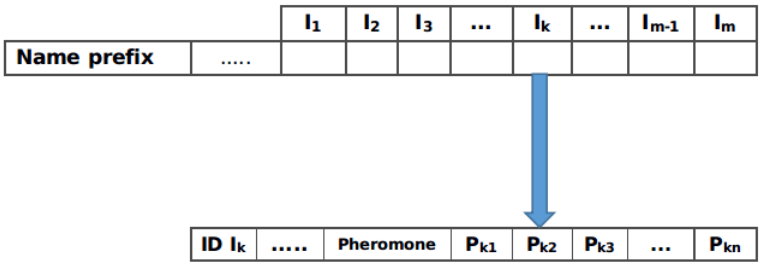


Figure 6.3: FIB entry data structure in **MC-QoS-FS**.

6.2.1.1 The moving rule

Let us assume that an Interest (ant) is located in the node x_i. N_{x_i} is a set of nodes connected to x_i except the node where the Interest comes from. The proposed forwarding strategy **MC-QoS-FS** can forward the Interest on upstream toward one node of N_{x_i}, if ($|N_{x_i}| \geqq 1$).

The **MC-QoS-FS** uses the pheromone amount $\tau_{i,j}$ of the interface that connects nodes x_i to x_j, which is stored in the field *pheromone* of FIB entry matching the Interest, to compute the probability of the node x_j as next hop. The probability of sending the Interest located in the node x_i via the interface j connected to the node x_j is mathematically written as follows:

$$P_{i,j} = \begin{cases} \frac{\tau_{i,j}}{\Sigma_{j \in N_{x_i}} \tau_{i,j}}, & \text{if } j \in N_{x_i} \\ 0, & \text{if } j \notin N_{x_i} \end{cases} \tag{6.2.1}$$

where the transition probabilities $P_{i,j}$ of the node x_i fulfill the constraint $\Sigma_{j \in N_{x_i}} P_{i,j} = 1$, $i \in [1, |X|]$. Initially, the pheromone amount is $\tau_{i,j} = \tau_0$.

It is worth to notify that the formula (6.2.1) is inspired from (5.2.1), which is proposed by [74] as moving rule for the ant TSP algorithm. Therefore, the formula (6.2.1) has been obtained by setting the parameters that control the relative importance of pheromone and visibility respectively to $\alpha = 1$ and $\beta = 0$. Since the visibility is the distance in terms of the QoS parameters measurements of the link between the current node and the next hop, which is included in the amount of pheromone computed, we set this parameter to $\beta = 0$. For the pheromone parameter, since $\beta = 0$, the value $\alpha = 1$ is naturally the best value, for forwarding the incoming Interest according to the amount of pheromone.

6.2.1.2 Pheromone update

As known, when the real ant returns back to its nest, it deposits a chemical pheromone on all the path. In ant colony based TSP algorithm, when all the ants finished a cycle, i.e. when all the cities had been visited by the ants, the ants start a new cycle for updating the pheromone amount.

In our proposal, when the ant finds the requested data, it returns back to the consumer node (starting point or nest), taking the same path traversed during the research phase but in opposite direction. Consequently, it crosses the same nodes but in reverse order. Therefore, at each node, it updates the amount of pheromone according to the quality of the path traversed between the food source, which can be a caching router (content store) or a producer, and the current node. So, the proposed forwarding strategy **MC-QoS-FS** updates only in the field $pheromone$ corresponding to the incoming interface of the ant in the FIB entry matching the data name handled by the ant. The quality of the path depends on the path QoS parameters measurements that the ant has collected: $P_1, P_2, P_3, ..., P_n$.

Let us consider that in the node x_i, an incoming Data packet (ant) matches the prefix P in the FIB table is arriving through the interface k. The amount of pheromone that the ant will deposit (update) on the correspondent field is computed as follows:

$$\tau_{x_i}(P,k)_(new) = (1-\rho)\tau_{x_i}(P,k)_(old) + \rho \times \Delta\tau_{x_i}(P,k) \tag{6.2.2}$$

where $\tau_{x_i}(P,k)$ is the pheromone amount of the interface k for the prefix P in the node x_i. $\Delta\tau_{x_i}(P,k)$ is the amount of pheromone that the ant will deposit on the interface k of the node x_i for the prefix P.

$$\Delta\tau_{x_i}(P,k) = \begin{cases} \frac{Q}{L_k}, & \text{if the ant is arriving from the interface } k \\ 0, & \text{else} \end{cases} \tag{6.2.3}$$

Q is a tuning parameter. L_k is the path distance in terms of QoS parameters measurements $P_1, P_2, P_3, ..., P_n$. As known, there are two kinds of QoS parameters: beneficial (positive) criteria, and expense (negative or cost) criteria. So in the formula that compute L_k, we use for a beneficial criterion the value P_i, but we use $\frac{1}{P_i}$for an expense criterion. Thus, we can write the formula that calculate L_k as:

$$L_k = \sum_{j=1}^{n} w_j \times r_{k,j} \tag{6.2.4}$$

where $r_{k,j} = P_{k,j}$, if $P_{k,j}$ is an expense criterion, $r_{k,j} = \frac{1}{P_{k,j}}$, if $P_{k,j}$ is a beneficial criterion. By replacing L_k par its values, equation (6.2.3) will be redefined as follows:

$$\Delta\tau_{x_i}(P,k) = \begin{cases} \frac{Q}{\sum_{j=1}^{n} w_j \times r_{k,j}}, & \text{if the ant is arriving from the interface } k \\ 0, & \text{else} \end{cases} \tag{6.2.5}$$

with $\sum_{j=1}^{n} w_j = 1$, and $w_1\, w_2\, w_3\, ..., w_n$ are the weighting constants that represent respectively the importance of the QoS parameters, $P_1, P_2, P_3, ..., P_n$.

Since the formula (6.2.5) groups different kinds of QoS parameters with different measurement units for beneficial and expense criteria, the amount of pheromone that the ant will deposit cannot

be computed by the equation (6.2.5) as it is. In other word, the values of QoS parameters need to be normalized.

In order to normalize such values, the Table 6.2 details how the QoS parameters measurements of available interfaces are organized in the FIB entry data structure, that was presented previously in Figure 6.3. In the proposed forwarding strategy **MC-QoS-FS** normalizes the QoS parameters mea-

	P_1	P_2	P_3	...	P_n
I_1	V_{11}	V_{12}	V_{13}	...	V_{1n}
I_2	V_{21}	V_{22}	V_{23}	...	V_{2n}
I_3	V_{31}	V_{32}	V_{33}	...	V_{3n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
I_m	V_{m1}	V_{m2}	V_{m3}	...	V_{mn}

Table 6.2: QoS parameters measurements of FIB entry interfaces

surements according the values of all interfaces. More specifically, **MC-QoS-FS** starts by updating the QoS parameters measurements of the interface k by the the collected values, then it computes the best value for each parameter among all the interfaces (see Table 6.2). The best value for the beneficial parameters is the maximum value of all interfaces:

$$V_{max_{P_j}} = \max_{i=1,m}(V_{i,j}) \tag{6.2.6}$$

while the minimum value among all interfaces is the best value for the cost parameters:

$$V_{min_{P_j}} = \min_{i=1,m}(V_{i,j}) \tag{6.2.7}$$

The beneficial (positive) criteria are normalized as: $r_{k,j} = \frac{V_{k,j}}{V_{max_{P_j}}}$, with $V_{max_{P_j}}$ the biggest value of the criterion j for all the interfaces and $V_{k,j}$ is the value of the criterion j for the interface k. The expense (negative or cost) criterion are normalized as: $r_{k,j} = \frac{V_{min_{P_j}}}{V_{k,j}}$, where $V_{min_{P_j}}$ is the smallest value of the criterion j for all the interfaces and $V_{k,j}$ is the values of the criterion j for the interface k. Equation (6.2.2) will be redefined as follows:

$$\tau_{x_i}(P,k)_(new) = (1-\rho)\tau_{x_i}(P,K)_(old) + \rho \times \frac{Q}{\sum_{j=1}^{n} w_j \times r_{k,j}} \tag{6.2.8}$$

6.2.2 Forwarding decision-making algorithms

In this section, we explain in details how the proposed forwarding strategy **MC-QoS-FS** reacts to the main events: reception of Interest and Data. Furthermore, we present the algorithms that follow upon reception of both packets.

6.2.2.1 After receiving an Interest

The proposed forwarding strategy **MC-QoS-FS** starts by checking the Content Store (CS) for data matching the incoming Interest name. If matched data exist then it will be sent on downstream via the incoming interface of the Interest and the Interest will be dropped as it is considered satisfied. If requested data is not found in the CS, it looks up in PIT table, if an entry matching the Interest name **MC-QoS-FS** adds the incoming interface to the requesting interfaces list, then it drops the Interest. When the requested data is found, a copy will be sent back via incoming interfaces of Interest. If the Interest name is not found in the PIT, **MC-QoS-FS** creates a new PIT entry and looks up for the Interest name in the FIB table. If no FIB entry matches the Interest name, the Interest is deleted. Otherwise, there is a FIB entry that matches the requested data name, **MC-QoS-FS** will compute the selection probability of each available interface according (6.2.1).

In order to get the best data delivery performance and avoid network instability, the proposed forwarding strategy **MC-QoS-FS** does not forward the incoming Interest via the interface that has the biggest probability. But, it uses these probabilities inputs for the roulette wheel selection algorithm which makes an oriented randomly selection corresponding to the interface probability. Consequently, the proposed forwarding strategy does not always send the incoming Interests via the interface that has the biggest probability (the biggest pheromone amount); however, it gives opportunity to other interfaces that have a small probability (pheromone), which ensures better traffic distribution over the network links.

Once the selection process is finished and the forwarding interface is selected, the proposed forwarding strategy **MC-QoS-FS** updates the PIT entry with the selected interface as outgoing interface and finally it sends the Interest. Algorithm 7 explains in details through a pseudo-code how our forwarding strategy works when a router receives a new incoming Interest.

Algorithm 7: After receiving an Interest for **MC-QoS-FS**

```
if ((Interest.name) is found in CS) then
    Forward(ant, k)
else
    if ((Interest.name) is found in PIT) then
        add(Requesting Faces list, k)
    else
        if ((Interest.name) is found in FIB) then
            Create new PIT entry (Interest.name, nonce, interface, k)
            *** Selection process ***
            for (each face K∈ FIB Entry) do
                Compute the probability of interface k according to Equation (6.2.1)
            Use the roulette wheel to select the outgoing interface S
            *** Forwarding process ***
            Update PIT entry with S as out interface
            Send(Interest, S)
        else
            Drop Interest
```

We assume that the worst-case time complexity of the search in CS, PIT and FIB is $O(sizeoftable)$. let consider the following notation:

- n_c: the size of the CS table
- n_p: the number of entries in the PIT table
- n_b: the number of entries in the FIB table
- n_f: the number of available interfaces

The worst-case time complexity of the **Algorithm 7** is $O(n)$, where $n = \max(n_c, \max(n_p, n_b + n_f))$.

6.2.2.2 After receiving a Data packet

When a Data packet (ant) is arriving at each router trough the interface s, the proposed forwarding strategy **MC-QoS-FS** looks up if the incoming data name exists in the PIT entries. If no matching entry is found, the data packet will be dropped. Whenever a PIT entry matching the data handled by the ant is found, **MC-QoS-FS** starts by computing the QoS parameters measurements. Such calculation dependent on the kind of QoS parameters. Then the forwarding strategy updates (saves) them, in the dedicated fields of $P_1, P_2, P_3, ..., P_n$ (see Figure 6.3) corresponding to the incoming interface s of the data in the FIB entry matching the incoming data, by using the informations collected by the ant. After that **MC-QoS-FS** computes the best values of each parameters, then it computes the pheromone amount that the ant will deposit and it updates the field *pheromone* dedicated for this goal as explained in section 6.2.1.2. Next, **MC-QoS-FS** updates the QoS parameters measurements handled by the data packet (ant) (Figure 6.2) about the path traversed from the source to the current node for QoS parameters $P_1, P_2, P_3, ..., P_n$. Finally, **MC-QoS-FS** sends a copy of the incoming Data packet via all the interfaces of the requesting interfaces list to the nodes on downstream, and stores the received data in CS. Once the Data packet is forwarded to downstream nodes, it deletes its PIT entry. Algorithm 8 details how our forwarding strategy works when a Data packet is received at each router.

The worst-case time complexity of the **Algorithm 8** is $O(n)$, where $n = n_p + n_b + n_f$.

6.3 Flow Congestion Reduction Ant colony based Quality of Service aware forwarding strategy (FCR-QoS-FS)

In this section we present, the design of a new forwarding strategy called Flow Congestion Reduction Ant Colony based Quality of Service aware Forwarding Strategy (**FCR-QoS-FS**), by using principles described in previous section. The proposed forwarding strategy **FCR-QoS-FS** is designed to be able, in addition to optimizing the QoS parameters taken into account in the chapter 5

Algorithm 8: After receiving a data packet

```
if ((Data.name) is found in PIT ) then
    Computing the QoS parameters measurements P_1, P_2, P_3, ...., P_n
    Updating the Fields of s in the FIB entry matching incoming data
    Computing the pherome amount as explained in 6.2.1.2
    Update the pheromone field by using the formula (6.2.8)
    Computing and updating the fields of the data packet
    Updating the QoS parameters measurements handled by the data packet
    CS.insert(Data)
    for (eachfaceK∈Requesting Faces) do
        Forwarder(Ant, K)
    Erase(PIT.entry)
else
    drop data End
```

(RTT, bandwidth, and cost), to adapt as much as possible to the network dynamic traffic, in particular to the problems of congestion. The main idea of **FCR-QoS-FS** consists on intelligently direct the flow to least congested links, which will decrease the amount of dropped packets. In order to achieve this goal, we have designed **FCR-QoS-FS** for taking into account other parameters that have a high and direct impact to congestion.

6.3.1 Designing of FCR-QoS-FS

In NDN, each Interest packet forwarded on upstream through an interface will receive at most one data packet from the same interface. Based on hop-by-hop and the symmetric packets flow (Interest-Data) NDN features, we introduce two other parameters that have a high and impact for congestion, which are IAR and DSR. Consequently, the **FCR-QoS-FS** takes into account five QoS parameters: cost, bandwidth (BW), *Round Trip Time* (RTT), *Interest Arrival Rate*(IAR) and *Data Satisfaction Rate* (DSR). Since these QoS parameters measurements will be used by **FCR-QoS-FS** to compute the pheromone amount that the ant will deposit, in the corresponding field of the incoming data interface in the FIB entry matching the received data name, all **FCR-QoS-FS** forwarding decisions for Interests and Data packets are based on the values of these parameters.

In this section, we detail how to adapt the node data structures and Data packet data structure to compute and update these parameters, and how the proposed forwarding strategy **FCR-QoS-FS** uses them in the forwarding decisions.

As explained in Section 6.2.1, to implement the proposed forwarding strategy **FCR-QoS-FS**, it is necessary to adapt the node data structures to **FCR-QoS-FS** design requirements. Consequently, we add other fields to the node data structure: one field is used to store the pheromone and the other to save the network QoS parameters metrics collected by the ants. The added fields are:

- **Pheromone**: it stores the amount of pheromone;

- **RTT**: it stores the round trip time elapsed between the departure of the ant and its return from the same interface;
- **BW**: it stores the path residual bandwidth traversed by the ant from the food source to the current node;
- **cost**: it stores the path cost traversed by the ant from the food source to the current node;
- **IAR** and **DSR** are used respectively, to store the Interest Arrival Rate and the Data Satisfaction Rate of the path traversed by the ant from the food source to the current node.

Figure 6.4 shows the added fields to FIB table data structure. They are used to compute the pheromone amount that the ant will depose. The details are described in Section 6.3.1.9.

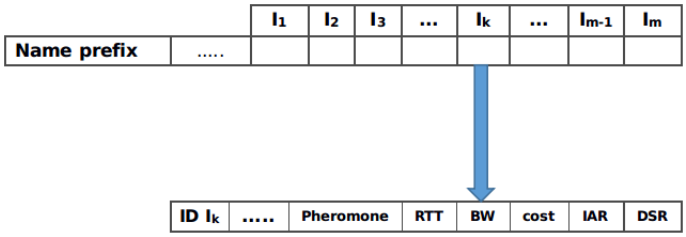


Figure 6.4: The FIB entry data structure in **FCR-QoS-FS**

It is also necessary to adapt the Data packet structure to **FCR-QoS-FS** design requirements. The added fields are:

- **BW**: it stores the path residual bandwidth traversed by the ant from the food source to the current node;
- **cost**: it stores the path cost traversed by the ant from the food source to the current node;
- **IAR** and **DSR**: they are used respectively, to store the Interest Arrival Rate and Data Satisfaction Rate of the path traversed by the ant from the food source to the current node;

Since the **RTT** is computed by a local timer at the level of router, no field is added to the Data packet Data structure. Figure 6.5 shows the added fields to Data packet structure.

Figure 6.5: The data structure Data packet in **FCR-QoS-FS**.

6.3.1.1 Criterion 1: Bandwidth

FCR-QoS-FS computes and updates the bandwidth QoS parameter exactly as explained in 5.3.1.1.

6.3.1.2 Criterion 2: Cost

FCR-QoS-FS computes and updates the cost QoS parameter as explained in 5.3.1.2

6.3.1.3 Criterion 3: Round Trip Time (RTT)

FCR-QoS-FS computes and updates the Round Trip Time (RTT) as explained in 5.3.1.3

6.3.1.4 Criterion 4: Interest Arrival Rate (IAR)

The *Interest Arrival Rate* (IAR) of the node X is the number of Interest packets received from all its interfaces per second. So, if $countI$ is the number of interests arrived to the node X from all its interfaces in the time interval D, the IAR of the node X can be computed as follows:

$$IAR(X) = \frac{countI}{D} \tag{6.3.9}$$

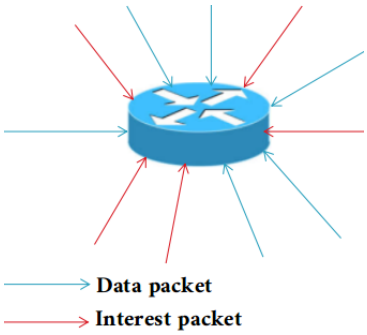


Figure 6.6: Packets (Interest and Data) received at a router

The IAR of the path connecting the food source node R_n to the current node R_i is the biggest IAR value of the nodes that form the path (R_i not included), which is equal the ant IAR field value. Such path corresponds to the one that was traversed by the ant received from the interface k, and handled the data corresponding to prefix P in FIB table. We can write the IAR of the path as follows:

$$IAR(R_i, P, k) = Ant.IAR \tag{6.3.10}$$

where:

$$Ant.IAR = \max(IAR(R_{i+1}), Ant.IAR) = \max(IAR(R_{i+1}), \max(IAR(R_{i+2}), Ant.IAR)) \tag{6.3.11}$$

By recursion :

$$IAR(R_i, P, k) = \max(IAR(R_{i+1}), IAR(R_{i+2}), ..., IAR(R_n)) \tag{6.3.12}$$

6.3.1.5 Criterion 5: Data Satisfaction Rate (DSR)

Before defining the Data Satisfaction Rate (DSR), it is necessary to define the Data Arrival Rate (DAR). The DAR of the node X is the number of Data packets received from all its interfaces per second. So, if $countD$ is the number of Data packets received by the node X from all its interfaces in the time interval D, the DAR of the node X is computed as follow:

$$DAR(X) = \frac{countD}{D} \tag{6.3.13}$$

The Data Satisfaction Rate (DSR) of the node X in the interval of time D is the rate of interests forwarded on upstream that have received a response. So, if DAR and IAR are respectively the Data and Interest Arrival Rate computed in the time interval D, the Data Satisfaction Rate of node X for the time interval D is computed as follow:

$$DSR(X) = \frac{DAR}{IAR} \tag{6.3.14}$$

The DSR of the path connects the food source router R_n to current router R_i, that was traversed by the ant received from the interface k, and handled the data corresponding to prefix P in FIB table, is the smallest DSR value of the nodes that form the path (R_i not included), which is the ant DSR field value. We can write it as follow:

$$DSR(R_i, P, k) = Ant.DSR \tag{6.3.15}$$

where:

$$Ant.DSR = \min(DSR(R_{i+1}), Ant.DSR) = \min(DSR(R_{i+1}), \min(DSR(R_{i+2}), Ant.DSR)) \tag{6.3.16}$$

By recursion :

$$DSR(R_i, P, k) = \min(DSR(R_{i+1}), DSR(R_{i+2}), ..., DSR(R_n)) \tag{6.3.17}$$

6.3.1.6 Computing Node IAR and DSR

In addition to the packets forward and backward process, **FCR-QoS-FS** executes in parallel another process for computing IAR and DSR, where it uses two counters $counterI$ and $counterD$ which are initially set to zero. The first one is used to compute the incoming interests from all interfaces of the node, and the second one is used to compute the incoming data also from all interfaces of the node. Since the routers are capable to make difference between Interest packets and Data packets, when an Interest or Data packet is received through an interface k at router X, the counter matching the kind of incoming packet at router X will be increment by one. The proposed forwarding strategy continues to compute the Interest and Data arrival number for the duration D which we fixed manually. Once the duration D is elapsed, it uses these counters values to compute IAR and DSR according the formulas 6.3.1.4 and 6.3.1.5. Then it resets the timer to zero and counters to zero.

6.3.1.7 Updating of the IAR and DSR path

While the ant (Data packet) returns back to its nest, the **FCR-QoS-FS** collects and updates QoS parameters measurements about the quality of the path that it traversed in terms of bandwidth, cost, IAR and DSR and stores them in the fields dedicated for this aim as additional information in the Data packet. Thus, when the Data packet arrives to each node, **FCR-QoS-FS** computes and updates bandwidth, cost and RTT as the same methods as explained in Chapter 5. The computation and update of IAR and DSR path according to the strategy are detailed.

We have explained in Section 6.3.1.6 how **FCR-QoS-FS** computes node IAR and DSR. These values are updated each time interval D. The IAR and DSR fields that are added to the Data packet data structure are used, as explained before, respectively to store the IAR and DSR values of the path traversed from the source node R_n to last node R_{i+1}.

In **FCR-QoS-FS**, the IAR used to represent one path is the worst node IAR of the nodes that form that path. This is represents the biggest IAR value of nodes form that path. Consequently, at each node, when a Data packet is arrived, **FCR-QoS-FS** updates Data packet IAR field by the current node IAR, if the current node IAR is bigger than incoming data packet IAR field, before forwarded the incoming packet to the next hop on downstream. The DSR of one path is the worst node DSR of the nodes that form the path, which is the lowest DSR value of nodes that form the path. Consequently, at each node, when a data packet is arrived, **FCR-QoS-FS** updates Data packet DSR field by the current node DSR, if the current node IAR is smaller than incoming data packet DSR field, before forwarded the incoming packet to the next hop on downstream.

It is worth to note that **FCR-QoS-FS** updates the IAR and DSR fields corresponding to the incoming interface of the Data packet in the FIB entry matching the incoming data name, respectively by the IAR and DSR fields values of the incoming Data packet. Thus, it computes the pheromone amount the ant will deposit according to these and updates the pheromone field. After that it updates

IAR and DSR data packet fields.

By using the parameters IAR and DSR, **FCR-QoS-FS** has a good idea about the real time state of the network. In order to get the best data delivery performance and avoid the instability of the network due to congestion problems, **FCR-QoS-FS** should forward the incoming interests through the interfaces that have a small IAR value and a big DSR value. Since in NDN the data packets take the same path of interests but in opposite direction, the interfaces that have a big IAR value, which connect to paths that have a big IAR, are exposed with high probability to be congested. Furthermore, the interfaces that have a small DSR, which are connected to the paths that have a small DSR, are more congested than other paths. In other words, whenever DSR is smaller, the path is more congested.

6.3.1.8 The moving rule

The proposed forwarding strategy **FCR-QoS-FS** uses the equation (6.2.1) defined in Section 6.2.1.1 to compute the probability of sending the Interest through the available interfaces.

6.3.1.9 Pheromone update

To present the **FCR-QoS-FS** process of updating the pheromone, we apply the general principle described in Section 6.2.1.2. Consequently, when in the node x_i, a Data packet (ant) matches the prefix P in FIB table, arriving trough the interface k, the amount of pheromone is updated according to the formula (6.2.8). Hence, the proposed forwarding strategy **FCR-QoS-FS** takes into account the QoS parameters cost, RTT, bandwidth, IAR and DSR, in order to calculate the pheromone amount that the ant will deposit is dependent on the values that it collected about the quality of path traversed in term of these parameters. In order to exploit the formula (6.2.8), with the QoS parameters used in the proposal, it is necessary to start by computing the best value of each parameter and normalize them next. So, since cost, RTT and IAR are expense criteria, they are normalized respectively as: $\frac{Bcost}{cost}$, $\frac{BRTT}{RTT}$ and $\frac{BIAR}{IAR}$, where: $Bcost$ is the min cost of all the interfaces and $cost$ is the cost of the interface k. $BRTT$ is the minimum RTT of all the interfaces and RTT is the RTT of the interface k. $BIAR$ is the minimum IAR of all the interfaces and IAR is the IAR of the interface k. Since *bandwidth* and *DSR* are a beneficial criteria, they are normalized respectively as follows: $\frac{BW}{BBW}$ and $\frac{DSR}{BDSR}$, where BBW is the maximum value of all the interfaces and BW is the bandwidth of the interface k. $BDSR$ is the maximum DSR of all the interfaces and DSR is the interface DSR. **FCR-QoS-FS** updates the pheromone amount $\tau_{x_i}(P,k)$ as follows:

$$\tau_{x_i}(P,K)_(new) = (1-\rho)\tau_{x_i}(P,k)_(old) + \rho \times \frac{Q}{c1 * \frac{BRTT}{RTT} + c2 * \frac{BW}{BBW} + c3 * \frac{Bcost}{cost} + c4 * \frac{BIAR}{IAR} + c5 * \frac{DSR}{BDSR}} \tag{6.3.18}$$

where $c1 + c2 + c3 + c4 + c5 = 1$, the parameters $c1, c2, c3, c4$ and $c5$ are the weighting constants that represent respectively the importance of the RTT, bandwidth, cost, IAR and DSR. Q is a tuning parameter set manually.

6.3.2 Forwarding decision-making algorithms

In this section, we explain in detail how **FCR-QoS-FS** reacts to the main events of Interest and Data reception, and we present the algorithms that follow such events.

6.3.2.1 After receiving an Interest

When an Interest packet is arriving to any node, the proposed forwarding strategy **FCR-QoS-FS** starts by incrementing the counter $countI$ used to compute the number of incoming interests. Then it performs the same steps that **MC-QoS-FS**, as presented in 6.2.2.1. But in addition to those steps, once the selection process is finished and the forwarding interface is selected, **FCR-QoS-FS** reserves the necessary bandwidth from the selected interface, then it updates the PIT entry with the selected interface as outgoing interface and finally it sends the Interest. Algorithm 9 explains in details through a pseudo-code how the proposed forwarding strategy **FCR-QoS-FS** works when a router receives a new incoming Interest.

Algorithm 9: After receiving an Interest

```
countI ++
if ((Interest.name) is found in CS) then
    Forward(ant, k)
else
    if ((Interest.name) is found in PIT) then
        add(Requesting Faces list, k)
    else
        if ((Interest.name) is found in FIB) then
            Create new PIT entry (Interest.name, nonce, interface, k)
            *** Selection process ***
            for (each face K∈ FIB Entry) do
                Compute the probability of interface K according to equation (6.2.1)
            Use the roulette wheel to select the out interface S
            *** Forwarding process ***
            BW_{int-S-res} ← BW_{int-S-res} − BW_{req}
            Update PIT entry with S as out interface
            Send(Interest, S)
        else
            Drop Interest
```

The worst-case time complexity of the **Algorithm 9** is $O(n)$, where $n = \max(n_c, \max(n_p, n_b + n_f))$.

6.3.2.2 After receiving a Data packet

When a Data packet (ant) is received by each router, the proposed forwarding strategy **FCR-QoS-FS** starts by incrementing the counter $countD$ used to compute the number of incoming Data packets, then looks up if the incoming data name exists. If no matching entry is found, the Data packet will be dropped.

After that **MC-QoS-FS** computes the best values of each parameters, then it computes the pheromone amount that the ant will deposit and it updates the field *pheromone* dedicated for this goal as explained in Section 6.2.1.2. Next, **MC-QoS-FS** updates the QoS parameters measurements handled by the Data packet (ant) (Figure 6.2) about the path traversed from the source to the current node for QoS parameters $P_1, P_2, P_3, ..., P_n$.

Whenever a PIT entry matching the data name is found, **FCR-QoS-FS** starts by releasing the reserved bandwidth from the incoming interface of the ant. Next it computes the QoS parameters (RTT, BW, cost, IAR and DSR) measurements of the path connecting the current node to the node from where the data was retrieved, by using the values collected by the ant, as explained previously. Furthermore, it updates (saves) them in the fields dedicated to this goal corresponding to the incoming interface of the data in the FIB entry matching the incoming data. After that, **FCR-QoS-FS** updates the QoS parameters (BW, cost, IAR and DSR) measurement handled by the ant about the path traversed.

Afterwards, it computes the pheromone amount that the ant will deposit and it updates (saves the values) in the fields dedicated for this goal as explained in Section 6.3.1.9.

Finally, **FCR-QoS-FS** sends a copy of the incoming data (ant) via all the interfaces of the requesting interfaces list to the nodes on downstream, and stores the received data in CS. Once the data is forwarded to downstream nodes, it deletes its PIT entry. Algorithm 10 details how the forwarding strategy **FCR-QoS-FS** works when a router receives a Data packet.

The worst-case time complexity of the **Algorithm 10** is $O(n)$, where $n = n_p + n_b + n_f$.

6.4 Performance Evaluation

In order to evaluate the proposed forwarding strategy named **FCR-QoS-FS**, ndnSIM simulator for ns3 has been used [71]. We have compared the performance of the proposed forwarding strategy **FCR-QoS-FS** with **AC-QoS-FS** that we presented in Chapter 5 and the Best Route forwarding strategy [70]. More specifically, we have analyzed the obtained results, by focusing on the same metrics as in Chapter 5 which are:

- The mean data delivery time, which represents the mean time elapsed between the sending of Interest and Data reception.

Algorithm 10: After receiving a data packet

```
countD++
if ((Data.name) is found in PIT ) then
    RTT(X,P,S) = Time_Receiving − Time_Forwarding
    costP(X,P,S) = Ant.cost() + cost(X,Y,S)
    IAR(X,P,S) = Ant.IAR
    DSR(X,P,S) = Ant.DSR
    IAR = Min(IARX, IAR(X,P,S))
    Ant.IAR(IAR)
    DSR = Max(DSRX, DSR(X,P,S))
    Ant.DSR(DSR)
    BW(X)_int−S−res = BW(X)_int−S−res + BW_req
    BW(X,P,S)_int−S−path = min(BW(X)_int−S−res, Ant.BW())
    Ant.BW(BW(X)_int−S−path)
    Ant.cost(costP)
    Update the pheromone by using the formula (6.3.18)
    CS.insert(Data)
    for (eachfaceK∈Requesting Faces) do
        Forwarder(Ant,k)
    Erase(PIT.entry)
else
    drop data End
```

- The mean cost, which represents the mean of the sum of links cost traversed by the Data packet from the data source to the consumer.
- The mean hops count, representing the mean number of hops traversed by the data between data source and the consumer.
- The dropped packets in Kilobyte.
- The hit ratio, which represents the ratio of the total number of cache hits over the total number of cache (misses and hits).

The performance of strategies are evaluated under Abilene topology composed of 12 routers as a core of the simulated network. In the simulation scenario each core router is attached to 1 repository used to store permanent contents and it is also attached to variable number of consumers between 2 and 4. Thus, the nodes total number is equal to 64. We have used a catalog of 100000 contents, where each content has 3 random replicas in the different repositories and each router can store up to 1000 chunks in its CS. To populate the FIB table of nodes we have used ***ndnGlobalRoutingHelper*** library provided by ndnSIM simulator. In this scenario, the consumers request contents according to the largely adopted Zipf's law. Concerning the contents popularity parameter α, we have considered three values (0.8; 1.0; and 1.2) [103] with different frequencies (3, 5, 7, 10, 15, 20, and 30 Interests per second).

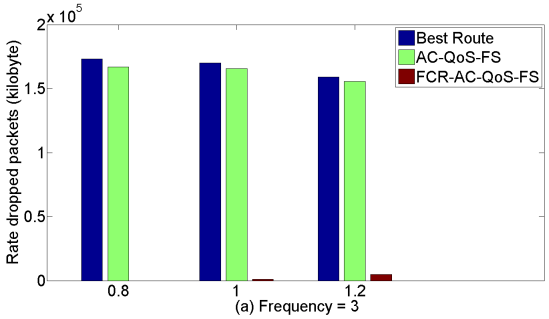
It is worth mentioning that the proposed forwarding strategy **FCR-QoS-FS** results have been obtained with the following configuration: $c1 = c2 = c3 = c4 = c5 = 1/5$ which means that RTT,

bandwidth, cost, IAR and DSR have the same importance for the application or the end user and we have also tuned $\rho = 0.2$ and $Q = 10$. While, the **AC-QoS-FS** results have been obtained with the following configuration: $c1 = c2 = c3 = 1/3$. The parameters RTT, bandwidth and cost also have the same importance. Furthermore we have set $\rho = 0.2$ and $Q = 10$.

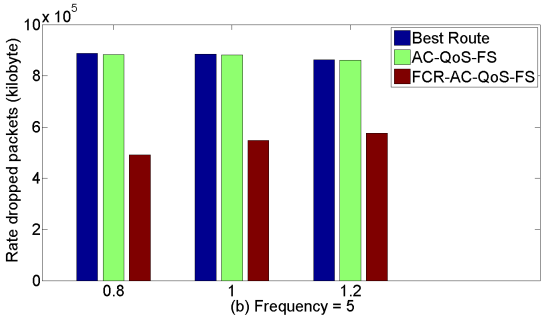
By analyzing the obtained results considering all the frequencies, and values of popularity parameter α mentioned above, we can notice that the Best Route forwarding strategy provides relative good performance when compared to **AC-QoS-FS** with regards to the mean hops count, cost and hit ratio Tables 6.3 and 6.4. As the principle used by Best Route approach is based on the minimizing cost and hop counts, it forwards the incoming Interests through the paths that have a lower cost and hop count, which explain the good obtained results in terms of RTT, cost and hop count comparing to our proposed forwarding strategies, But it provides the worst performances concerning to the amount drop packets (ADP).

It is worth mentioning that the obtained results for cost, hops count and RTT are measured only for the packets received by the consumer, so the dropped packets are not took into account (the cost and RTT for the dropped packets are not measured), thus if we consider these two parameters for the important amount of the dropped packets for Best Route, we can ultimately conclude that our proposals (especially **FCR-QoS-FS**) offer the best performance. The obtained results show that the **AC-QoS-FS** provide better performance comparing to **FCR-QoS-FS**, for cost, hops count and hit ratio. Concerning RTT, the obtained results are comparable with a slightly better preference for **FCR-QoS-FS**. In terms of the amount dropped packets, **FCR-QoS-FS** provides the best performance. As mentioned above, since the cost, hops count and RTT are measured only for the packets received by the consumers, and regarding to expressive difference concerning the amount of dropped packets, it is clear that the **FCR-QoS-FS** is more efficient than the **AC-QoS-FS**. Since the difference between the **AC-QoS-FS** and **FCR-QoS-FS** consists only that this latter takes into account the IAR and DSR parameters, the addition of these parameters leads to the minimizing of the amount of dropped packets. Based on the values of these two parameters the forwarding strategy is able to direct packets to the less congested paths.

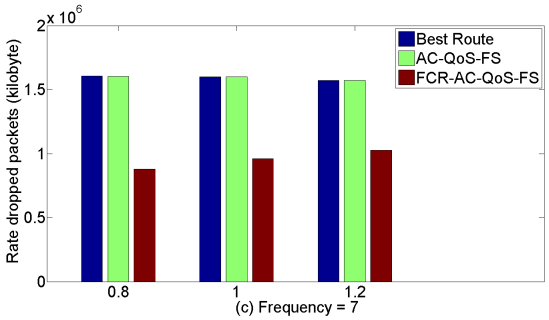
Figure 6.7 demonstrates that **FCR-QoS-FS** is more effective than **AC-QoS-FS** and Best Route with regards to the amount of dropped packets in kilobyte and Table 6.5 demonstrates in details the rate (in $\%$) of the amount of dropped packets for **FCR-QoS-FS** against **AC-QoS-FS** and Best Route. For example, when the frequency is 3 requests per second and the popularity parameter $\alpha = 0.8$the **FCR-QoS-FS** does not drop any packet, while **AC-QoS-FS** and Best Route drop respectively 167005 and 173120 Kilobytes, **FCR-QoS-FS** drop (-100$\%$) against **AC-QoS-FS** and **Best Route**. However, when the frequency is 15 requests per second and the popularity parameter $\alpha = 1.2$the **FCR-QoS-FS** drops 1239498 Kilobytes, while **AC-QoS-FS** and Best Route drop respectively 1754740 and 1755011 Kilobytes, **FCR-QoS-FS** drops (-29,362$\%$) against **AC-QoS-FS** and (-29,373$\%$) **Best Route**.



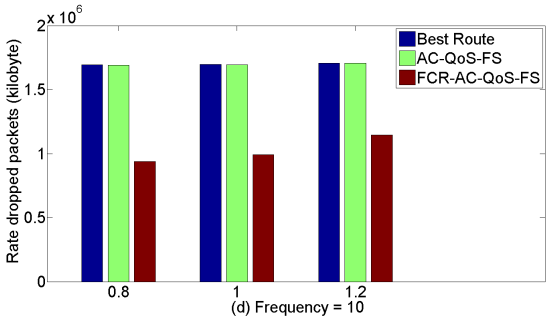
((a)) Frequency=3



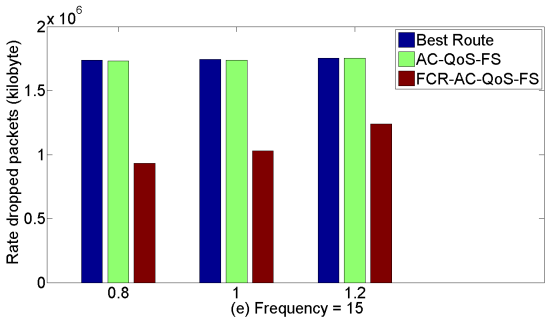
((b)) Frequency=5



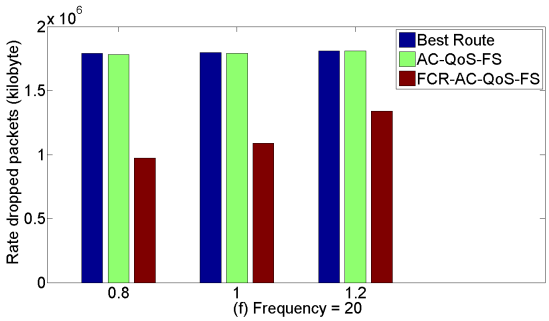
((c)) Frequency=7



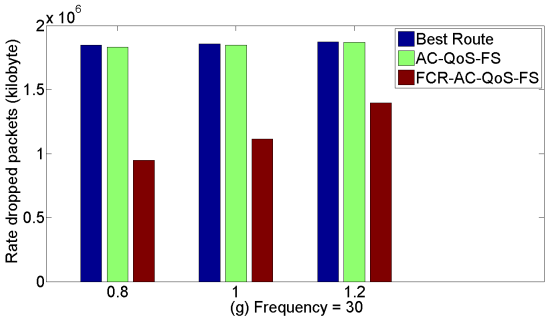
((d)) Frequency=10



((e)) Frequency=15



((f)) Frequency=20



((g)) Frequency=30

Figure 6.7: Dropped packets in kilobyte

We can particularly notice that for increasing frequencies, the amount of dropped packets is increasing also for all strategies. Furthermore, the values of RTT, cost and hops count decrease with the increasing value of the parameter α for all frequencies, and the values of these metrics also decreases with the increasing of used frequencies. Finally, we can remark that RTT of the **FCR-QoS-FS** becomes better when compared to other strategies RTT, for the frequencies 20 and 30. We point out that, it is clear that the obtained results have provided very good performances, but it is possible to improve them, by fine tuning of ($c1, c2, c3, c4, c5, \rho$ and Q) parameters according to the application or end user requirements.

Concerning the convergence of the proposed solution, it is worth noting that the Internet network is by its nature highly dynamic and our proposal forward strategy makes its forwarding decision based on the real-time QoS parameters collected from the network. In this direction, we aimed to compare our proposals with other solutions under the same network conditions, although we are not able to obtain a steady-state topology (due to the inherent dynamicity of the system). The simulations were then performed for the same topologies and time durations (720s) several times, as to obtain averages from hundreds of thousands of served Interests. This allowed us to validate all strategies.

6.5 Conclusion

This chapter presents at first the **M**ulti **C**riteron based **A**nt **C**olony **Q**uality-of-**S**ervice (QoS) **F**orwarding **S**trategy (**MC-QoS-FS**) which is a designing model of forwarding strategy based on Ant Colony, which we have proposed to handle the issue of routing with QoS in NDN. This latter offers operators the opportunity of designing a forwarding strategy that enable to satisfy the applications or users requirements through the possibility of taking into account n QoS parameters from different kind. After that, we have designed Flow Congestion Reduction Ant Colony based Quality of Service aware forwarding strategy (FCR-QoS-FS) by applying the model designing principles proposed in **MC-QoS-FS**. **FCR-QoS-FS** takes into account five QoS parameters: cost, RTT, bandwidth, interest arrival rate (IAR) and data satisfaction rate (DSR). So, in addition to the optimization of cost,BW, and RTT presented in Chapter 5, **FCR-QoS-FS** takes into account two other parameters: IAR and DSR, which have a high and direct impact on congestion. The performance evaluation, in different simulation scenarios in ndnSIM simulator for ns3 under various requests frequency for different values of popularity parameter, demonstrates the effectiveness of our new solution, which gives the best results against **AC-QoS-FS** (presented in Chapter 5) and Best Route. This is due to the lower of the amount packets dropped in Kilobytes for **FCR-QoS-FS** when compared to the two other strategies. Furthermore, the obtained results demonstrate that we could design a forwarding strategy that ensure the best performances by tuning ($c1, c2, c3, c4, c5, \rho$ and Q) parameters according to the application or end user requirements.

		RTT			Hop count			Cost		
Freq	Alpha	AC-QoS-FS	FCR-AC-QoS-FS	Best Route	AC-QoS-FS	FCR-AC-QoS-FS	Best Route	AC-QoS-FS	FCR-AC-QoS-FS	Best Route
3	0,8	**1,97573526**	1,993076988	1,989417711	3,371342913	3,89802502	**2,970577369**	1147,038526	1594,122263	**912,9105001**
	1	1,958529924	1,987179811	**1,953675941**	2,895580303	3,313238028	**2,549725841**	941,3606178	1302,647674	**877,2088462**
	1,2	1,963672817	**1,955538413**	1,965266216	2,318687105	2,675342697	**2,000149082**	**655,3169978**	947,5159252	850,3550378
5	0,8	1,994130135	**1,972369169**	1,998667947	3,337905102	3,979088158	**2,955548885**	1127,13443	1654,19891	**879,9650587**
	1	**1,966020391**	1,994463567	1,99109642	2,982858257	3,453649869	**2,638352392**	973,1002286	1367,607938	**862,2012903**
	1,2	1,946001514	**1,943288811**	1,958984037	2,513579799	2,888796768	**2,212929687**	**747,8107376**	1060,447011	843,7990347
7	0,8	1,994308428	1,997646164	**1,975349177**	3,355826856	3,931244234	**2,961673612**	1127,162695	1623,572732	**902,3824647**
	1	**1,943880911**	1,963731862	1,966320877	3,023379619	3,503201384	**2,679954238**	995,6118218	1394,29951	**862,0376304**
	1,2	1,937806409	**1,934536862**	1,945752836	2,615009156	3,003910457	**2,319378524**	**796,376351**	1131,705951	833,9306451
10	0,8	1,987357483	1,985347526	**1,97816393**	3,328918184	3,912803336	**2,916977817**	1119,314525	1608,727826	**835,7718184**
	1	1,970995736	1,972643837	**1,968903278**	3,036923766	3,576067731	**2,692675701**	994,5347734	1428,613431	**817,8930153**
	1,2	**1,929277275**	1,941114516	1,943608005	2,674611376	3,091997891	**2,392733581**	825,739397	1172,741663	**798,9684726**
15	0,8	1,993736636	1,982281866	**1,971700053**	3,410135281	3,982176683	**3,003593807**	1154,690212	1634,789451	**890,9375491**
	1	1,958480351	1,974273165	**1,955271116**	3,10710739	3,662941817	**2,764442306**	1026,766341	1469,490659	**874,5013542**
	1,2	**1,91692727**	1,919130214	1,917069115	2,768381635	3,195572064	**2,495224412**	863,0286477	1232,015579	**846,9767855**
20	0,8	1,97704662	**1,975080125**	1,988142976	3,40944507	3,990379102	**2,985946075**	1148,825493	1634,219732	**844,6642914**
	1	1,949302945	**1,944724438**	1,948004573	3,144233569	3,695494777	**2,756941571**	1047,595202	1489,909178	**823,6024561**
	1,2	1,911015907	**1,900878962**	1,903307012	2,823922758	3,264028374	**2,518965101**	892,7403314	1264,291562	**801,0979766**
30	0,8	1,97031307	**1,967128345**	1,981097937	3,442292944	4,038178018	**3,012102905**	1167,734258	1675,621956	**835,2276508**
	1	1,972679501	**1,961314332**	1,967963362	3,182069536	3,76144901	**2,821551154**	1064,838277	1526,924605	**870,5812642**
	1,2	1,844877696	**1,847351063**	1,862818196	2,88850806	3,370509412	**2,604834887**	922,0908978	1328,782023	**860,2226015**

Table 6.3: Obtained results

		ADP			Hit Ratio		
Freq	Alpha	AC-QoS-FS	FCR-AC-QoS-FS	Best Route	AC-QoS-FS	FCR-AC-QoS-FS	Best Route
3	0,8	167005	**0**	173120	7,44819448	5,87209802	**8,22246293**
	1	165780	**1058**	170151	14,12883067	10,89447032	**17,46674749**
	1,2	155561	**4924**	159044	18,32693647	13,62132168	**22,61122702**
5	0,8	883658	**491286**	888253	7,134466033	5,661459937	**7,798470393**
	1	881823	**548062**	884167	10,44918422	7,927880845	**12,70758177**
	1,2	861972	**576777**	863276	13,58967503	10,90034868	**16,8489191**
7	0,8	1602910	**878104**	1607179	7,337815098	5,807749548	**7,740230109**
	1	1599212	**961636**	1601801	9,044984865	7,071747824	**9,791068194**
	1,2	1570602	**1027289**	1570890	12,28775376	10,30874447	**14,92679935**
10	0,8	1691139	**937955**	1695177	6,708567575	5,290423423	**6,940934323**
	1	1695024	**992953**	1698245	7,578050634	5,881549739	**6,940935581**
	1,2	1706777	**1147257**	1707456	11,15459836	9,413066351	**13,37006065**
15	0,8	1732857	**931352**	1739171	5,631657804	4,446701959	**6,043907155**
	1	1738821	**1030316**	1743140	6,63872717	5,587744946	**7,101773533**
	1,2	1754740	**1239498**	1755011	11,21407799	9,7632086	**13,20179346**
20	0,8	1782413	**972353**	1790627	4,814455664	3,998235444	**5,074972722**
	1	1791823	**1088929**	1797063	6,262078361	5,295615222	**6,821424647**
	1,2	1809847	**1340303**	1810541	11,41519397	9,987457857	**13,11848629**
30	0,8	1833606	**948374**	1848179	**3,68337756**	3,292612347	3,578480826
	1	1847275	**1114329**	1857821	5,433344157	4,968281424	**5,610792453**
	1,2	1869235	**1397760**	1872133	11,09087348	10,30679692	**12,72825921**

Table 6.4: Obtained results

Frequency	alpha	**AC-QoS-FS**	**Best Route**
3	0.8	-100 %	-100 %
	1.0	-99.3618048015442 %	-99.3781993640942 %
	1.2	-96.8346822146939 %	-96.9040014084153 %
5	0.8	-44.4031514454687 %	-44.6907581511124 %
	1.0	-37.8489787633119 %	-38.0137462719147 %
	1.2	-33.0863415516977 %	-33.1874163071833 %
7	0.8	-45.2181345178457 %	-45.3636464886612 %
	1.0	-39.8681350565153 %	-39.9653265293254 %
	1.2	-34.592659375195 %	-34.6046508667061 %
10	0.8	-44.5370841781781 %	-44.6691997354849 %
	1.0	-41.4195315228575 %	-41.5306389831856 %
	1.2	-32.7822556783927 %	-32,8089860002249 %
15	0.8	-46.2533838626038 %	-46.4485090885255 %
	1.0	-40.7462872831649 %	-40.8931009557465 %
	1.2	-29.3628685731219 %	-29.3737760048228 %
20	0.8	-45.4473794793911 %	-45.6976243516936 %
	1.0	-39.227870163515 %	-39.4050737230693 %
	1.2	-25.94385050228 %	-25.9722370274962 %
30	0.8	-48.2782015329356 %	-48.6860309526296 %
	1.0	-39.6771460665034 %	-40.0195713149975 %
	1.2	-25.2228852979962 %	-25.3386377997717 %

Table 6.5: The rate (in %) of the amount of dropped packets

Chapter 7

Conclusion

Contents

7.1 Summary of Contributions 135
7.2 Future work 137
7.3 Publications 139

In this chapter we present a general conclusion of this dissertation by summarizing our contributions and describing some perspectives for future works.

7.1 Summary of Contributions

The objective of this dissertation was the investigation of the routing issue in the networks designed according to the ICN paradigm. After a study of the basis and principles of various ICN architectures that are summarized in Chapter 2, we have decided that our contributions in this topic should be according to the NDN network architecture. More specifically, it includes the QoS-based routing which has just started in the ICN architectures, although, it has been widely discussed in the literature for the IP architecture.

Since the forwarding plane by means of its forwarding strategy module is the responsible of packets forwarding process, and routing is responsible of that populates the FIB and gives the information for helping the forwarding plane to take the best decisions as detailed in Chapter 3, all our three contributions are related to the design and implementation of three new adaptive forwarding strategies with QoS.

In Chapter 4, we have presented our first contribution which is a simple and adaptive QoS forwarding strategy called **QoS-FS**, that we proposed to ensure QoS-based routing in NDN. The main idea of **QoS-FS** is to monitor and estimate, in real-time, the QoS parameters of interfaces,

and integrate them in the decision making process of forwarding, which is adaptive to the network conditions and preferences of user. Two QoS parameters have been considered, namely bandwidth and Round Trip Time (RTT). The **QoS-FS** starts by a exploration phase for a time period fixed manually before shifting to the exploitation phase. The simulations results, under the ndnSIM simulator, showed that the proposed **QoS-FS** offers the users a better QoS in terms of delivery time than the Best Route and it offers also the smallest hops count when a traffic increases. This explains the importance of taking into account the real time network state in terms of QoS parameters.

In Chapter 5, we have presented our second contribution, which also consist on another adaptive forwarding strategy with QoS called Ant Colony based QoS-aware Forwarding Strategy (**AC-QoS-FS**). Based on the similarities between the natural behavior of real ants and NDN forwarding process, the ant colony algorithm optimization is adapted for designing and implementing **AC-QoS-FS**. This latter considered three QoS parameters: RTT, bandwidth, and cost. So, **AC-QoS-FS** does not have exploration and exploitation phases. The main idea of this strategy consists on collecting the real time network state about the three QoS parameters by using the data packets. Then **AC-QoS-FS** transforms these values to pheromone in order to use it for forwarding the future incoming Interest.

The obtained results under different simulation scenarios that we performed to evaluate the **AC-QoS-FS** performance show its effectiveness. Furthermore, by tuning the parameters related to algorithms proposed in our strategy, according to the application or end user requirements, we can ensure the best performances.

In the third contribution that is presented in Chapter 6, we start by introducing a designing model of adaptive forwarding strategy with QoS according to the ant colony optimization algorithm, to handle the issue of routing with QoS in NDN network. This model is a generalization of the approach suggested in the second contribution to support several QoS parameters. This model is called Multi Criterion Ant Colony QoS based Forwarding Strategy for NDN (**MC-QoS-FS**). The goal is to provide to operators a model that allows them to design and implement an adaptive forwarding strategy with QoS according to the considered QoS parameters (number and kind). Furthermore, the satisfaction process of applications or users requirements is possible by tunning handedly the algorithms parameters.

After that, by applying the suggested designing model **MC-QoS-FS** principles, we have designed a new adaptive forwarding strategy with QoS called Flow Congestion Reduction Ant Colony based Quality of Service aware forwarding strategy (**FCR-QoS-FS**). Five QoS parameters have been considered: cost, RTT, bandwidth, Interest Arrival Rate (IAR) and Data Satisfaction Rate (DSR). So, **FCR-QoS-FS**, in addition to the QoS parameters that are considered by **AC-QoS-FS** proposed in the second contribution, it takes into account also two other parameters: IAR and DSR, which represent respectively the real time flow state of Interests and Data packets. Since the values of these parameters are integrated in **FCR-QoS-FS**, they will have a high and direct impact on

congestion. The main idea of **FCR-QoS-FS** consists on intelligently direct the flow toward lowest congested links by using the indicator values IAR and DSR, which will decrease the amount of dropped packets.

We have simulated our different scenarios in ndnSIM simulator under various requests frequencies for different values of popularity a parameter that we performed to evaluate the performance of the forwarding strategies **FCR-QoS-FS**, **AC-QoS-FS** and Best Route strategies. The obtained results demonstrates the effectiveness of **FCR-QoS-FS**. The latter gives the best results when compared to **AC-QoS-FS** and Best Route, because the amount of packets dropped in Kilobytes of **FCR-QoS-FS** is lower than **AC-QoS-FS** and Best Route.

Furthermore, it is possible to get the best delivery performance by tuning the parameters values related to the proposed algorithms according to the applications or end users requirements.

7.2 Future work

From this work, many perspectives can be formulated, that we propose as follows:

- We have proposed in Chapter 6 a designing model of forwarding strategy based on Ant Colony Optimization Algorithm, which is able to handle the issue of routing with QoS in NDN. So, the algorithms developed and the described principles in the model can be considered as general procedures that can be used by the operators in order to offer the opportunity of designing the appropriate forwarding strategy corresponding to the applications or users requirements. This could be ensured by choosing different kinds of considered QoS parameters and their number, and by modifying also their importance weights. Consequently, we note as a first perspective, the exploration of our proposal by using different kind and number of QoS parameters and also by tuning their importance weights.

- In order to confirm the effectiveness of our proposed forwarding strategies, we should investigate their performances evaluation by testing them in the natural scenario when they work simultaneously with the available routing protocols such as NLSR in simulation mode under ndnSIM simulator. Unfortunately until now, the NLSR version on ndnSIM does not exist, but it will be available in the future.

- We can also notice as perspective the extension of our proposed forwarding strategies to support different network failures such as: link failure, prefix hijack, etc. In addition, we can investigate the adaptation of the proposed forwarding strategies to support different caching policies in order to reach the best effectiveness.

- Another perspective point that we suggest consists on the investigation of other optimization algorithms to use them in order to handle the QoS-based routing and the smart forwarding of

packets over multipaths, while considering QoS parameters aiming to reach more effectiveness.

- We plan to evaluate the performances of our proposals in a real testbed. For example under a NDN-based testbed by using Banana Pi routers.
- It is noteworthy that the contributions of this dissertation are considered among the first work in literature that address routing with QoS issue in ICN networks, particularly in NDN network. However, until now and to the best of our knowledge, there is no work that address the routing with quality of experience (QoE) issue. Therefore, this issue could be investigated.

7.3 Publications

Hereafter, we summarize the publications that have been elaborated during this thesis.

- A. Kerrouche, M. R. Senouci, and A. Mellouk, "QoS-FS: A new forwarding strategy with QoS for routing in named data networking," in 2016 IEEE International Conference on Communications (ICC), Institute of Electrical and Electronics Engineers (IEEE), may 2016.
- A. Kerrouche, M. Senouci, A. Mellouk, Thiago Abreu " AC-QoS-FS: Ant colony based QoS-aware forwarding strategy for routing in Named Data Networking" in 2017 IEEE International Conference on Communications (ICC), Institute of Electrical and Electronics Engineers (IEEE), Paris-France may 2017.
- A. Kerrouche, M. Senouci, A. Mellouk, Thiago Abreu "Multi-Criteria QoS-based Forwarding Strategy for NDN" to be submitted to an International Journal.

Chapter 8

Version Française Abrégée

Contents

8.1 Contexte Général de la thèse 141
8.2 Problématique 143
8.2.1 Routage et acheminement des paquets dans NDN 144
8.2.2 Routage avec Qualité de Service 145
8.2.3 Formulation du Problème 145
8.3 Nos contributions 146
8.3.1 QoS-FS: Stratégie d'acheminement basée sur la QoS pour NDN 146
8.3.2 AC-QoS-FS: Stratégie d'acheminement avec qualité de service basée sur les colonies de fourmis 148
8.3.3 MC-QoS-FS: Stratégie d'acheminement basée sur multi critères de qualité de service 150
8.4 Conclusion 155

8.1 Contexte Général de la thèse

L'Internet a connu un grand développement depuis les années 1960 jusqu'à nos jours. Au début, il était imprévisible que l'Internet aurait une telle croissance. En effet, dans les années 1970s Internet était un réseau académique appelé ARPANET [1], qui a été conçu pour répondre aux besoins de la période, comme le partage de ressources rares et coûteuses, tel que des périphériques, des ordinateurs centraux et des liaisons de communication longue distance. Il a également été utilisé pour la transmission de paquets de données entre un nombre limité de machines fixes [2]. En conséquence, l'architecture conçue a été centrée sur l'hôte. En d'autres termes, basée sur un modèle de communication qui permet la connexion entre deux machines (une adresse source et une adresse destination).

De nos jours, l'Internet est devenu indispensable dans notre vie quotidienne,. Le nombre d'appareils connectés à Internet a atteint plus de quatorze milliards en 2014, le nombre d'utilisateurs d'Internet a atteint plus de trois milliards, et des milliards de pages Web indexées sont disponibles [3]. Cette extraordinaire croissance d'Internet est accompagnée d'un changement significatif dans l'utilisation. Plus précisément, Internet a été conçu à l'origine pour permettre la communication entre les hôtes. Aujourd'hui, il est essentiellement utilisé pour la distribution et la récupération de contenu, compte tenu de la quantité d'informations stockées dans différents serveurs installés à travers le monde, provoquant un échange massif à travers le réseau Internet. En outre, les utilisateurs sont généralement intéressés par le contenu lui-même plutôt que par son emplacement. Ce changement d'usage du paradigme axé sur l'hôte vers le contenu a motivé l'émergence de nombreuses tentatives pour adapter la distribution de contenu à l'infrastructure Internet actuelle. Essentiellement, ces tentatives essayent de découpler le contenu des hôtes. Les tentatives antérieures telles que les réseaux peer to peer (P2P) et les Content Delivery Network (CDN) découplent le contenu et les hôtes au niveau de la couche application. De nombreuses études montrent que le P2P et le CDN peuvent améliorer la Qualité de Service (QdS) et la Qualité de l'Expérience (QdE) des utilisateurs finaux ; mais ils entraînent des coûts élevés et peuvent conduire à des solutions inefficaces.

Aussi, la croissance significative d'Internet et l'introduction de nouvelles applications pour répondre aux besoins émergents des utilisateurs ont conduit à l'apparition de nouvelles exigences architecturales telles que la mobilité, la sécurité, l'ubiquité, etc. En réalité, Internet n'a jamais été conçu pour répondre à ces exigences. Ainsi, afin d'aider l'évolution d'Internet, diverses solutions temporaires ont commencé à apparaître ; cependant, la plupart de ces solutions augmentent considérablement la complexité de l'architecture globale [1]. Les problèmes architecturaux sont classés selon [1] comme suite :

- Les problèmes à court terme, qui consistent en spam, sécurité, déni de service et le déploiement d'applications ;
- Les problèmes à moyen terme, qui consistent en le contrôle de la congestion, le routage inter-domaine, la mobilité, le multi-homing et l'ossification (conformité) [] architecturale due au changement dans l'utilisation de l'Internet, d'un réseau centré-machine à réseau centré-contenu ;
- Les problèmes à long terme, qui consistent en l'épuisement de l'espace d'adressage.

Dans la littérature, il y a deux approches essentielles pour résoudre ces défis. La première approche est la recherche incrémentale ou évolutive, ce qui signifie qu'un système étudié est changé progressivement et déplacé d'un état à un autre en soutenant le vieux potentiel et en ajoutant de nouvelles opportunités. Cela peut être réalisé par la compréhension du comportement de l'Internet actuel et l'identification des problèmes existants et émergents, afin de les résoudre [4]. La seconde

approche, “clean-slate”, signifie que chaque nouvel état du système est développé à partir de zéro, offrant de vieilles possibilités sur de nouveaux principes [5], en concevant une nouvelle architecture pour future Internet qui soit meilleure que l'architecture existante, en termes de sécurité, mobilité, résilience et d'autres propriétés sans être limitée par l'Internet actuel [4].

Grand nombre des chercheurs sont convaincus qu'il est impossible de résoudre les problèmes architecturaux mentionnés précédemment avec l'architecture Internet actuelle [4] [5]. En conséquence, de nombreuses contributions sont apparues dans le sens de repenser les hypothèses fondamentales et les décisions de conception et de partir de zéro. La communauté scientifique d'Internet propose aujourd'hui de se concentrer sur cette approche pour trouver la solution appropriée qui répond à toutes les problématiques plutôt que sur l'approche évolutive [5].

Dans ce contexte, Information Centric-Networking (ICN) est un nouveau paradigme qui présente une approche pour une nouvelle architecture du futur réseau Internet, centrée sur le contenu. Il est inspiré par le fait que l'utilisation d'Internet a été changée vers la distribution et la récupération de contenu au lieu de la communication entre les hôtes. L'approche ICN est actuellement considérée comme une solution prometteuse pour l'émergence d'une nouvelle architecture du futur Internet.

Plusieurs architectures ont été proposées par différents projets lancés dans le monde tels que : DONA, PURSUIT, SAIL, COMET, CONVERGENCE, CCN, NDN, etc. Parmi ces architectures, une d'entre elles se détache nettement, il s'agit de l'architecture dite NDN (Named Data Networking) [6], [7], [8], [9], [10] qui est aujourd'hui la plus avancée. Elle a été proposée par le centre PARC (Palo Alto Research Center) aux États-Unis, dans le cadre du programme de recherche portant sur une architecture du futur Internet (projet financé par l'organisme américain NSF, National Science Foundation). Cette architecture récente est aujourd'hui le centre d'attention de la communauté scientifique et plusieurs travaux lui sont consacrés.

8.2 Problématique

Étant donné que les architectures proposées selon le paradigme Information Centric-Networking sont très récentes, les recherches dans ce domaine sont à leurs débuts. Ainsi, divers défis de recherche doivent encore être abordés tels que la nomination, la sécurité, la résolution de noms, le routage, le stockage dans le réseau, la mise en cache, la mobilité, etc. [11]. Par conséquent, le succès de toute architecture ICN dépend de la qualité des solutions proposées pour chacun de ces défis de recherche. Dans cette thèse, nous avons étudié le problème de routage dans les réseaux ICN, à travers une architecture de type NDN (Named Data Networking). Par conséquent, nos contributions seront conçues et mises en œuvre selon l'architecture NDN, et seront également validées par des simulations en utilisant le simulateur ndnSIM (le simulateur de référence dans le domaine NDN).

L'Internet se caractérise par l'hétérogénéité de ses liens et les conditions de circulation dynamiques. Cela nécessite de prendre en compte la qualité du service dans le réseau et en particulier

dans le routage pour satisfaire les exigences des utilisateurs. Ce problème a été largement discuté dans la littérature pour l'architecture IP (Internet Protocol), alors qu'il vient de démarrer dans les architectures de type ICN.

Dans le réseau NDN, la livraison des paquets de données est également organisée comme dans l'architecture IP, sur deux plans (plan routage et plan d'acheminement). Cependant, le plan d'acheminement, au moyen de son module de stratégie d'acheminement est responsable de la transmission des paquets à travers les multiples chemins disponibles, et le plan de routage est utilisé uniquement pour aider le plan d'acheminement. En conséquence, nous nous sommes intéressés à cette thèse en améliorant la qualité de service perçue par les utilisateurs, en concevant et en mettant en œuvre des stratégies d'acheminement intelligentes capables de transmettre les paquets sur les multiples chemins tout en considérant les paramètres de QdS de l'état du réseau en temps réel.

8.2.1 Routage et acheminement des paquets dans NDN

Le plan de routage dans NDN sert le même but comme dans les réseaux IP : les fournisseurs de contenu annoncent les préfixes de noms et les autres routeurs calculent les chemins. En effet, le plan de routage est responsable de remplir les tables FIB des routeurs et de les mettre à jour. Dans les réseaux IP, le plan de routage est le plan de contrôle, qui est statefull, intelligent et adaptatif. Il effectue tout le travail, mais le plan d'acheminement n'est pas intelligent et le trafic est à sens unique. Les routeurs dans l'architecture IP transmettent les paquets strictement en suivant la table FIB via une interface unique [22]. Par conséquence, la fourniture robuste de paquets dans les réseaux IP dépend uniquement du plan de routage. Cependant, dans NDN, le plan d'acheminement est le plan de contrôle. En effet, le module de stratégie d'acheminement est capable de prendre des décisions d'acheminement tout seul. Il est statefull, bidirectionnel, intelligent et adaptatif. En outre, il est capable de prendre en charge une partie de la responsabilité du plan de routage [32]. Par conséquence, la conception du plan de routage NDN peut être différente de l'actuel plan de routage existant, car elle contribue uniquement à la construction de la table FIB.

Il est clair que la grande différence entre les réseaux IP et NDN consiste dans le plan d'acheminement. En effet, le plan d'acheminement, grâce au module stratégie d'acheminement est le secret de la puissance de NDN, peut détecter et récupérer des défaillances de réseau tout seul sans se reposer sur le plan de routage. Par conséquence, chaque routeur NDN est capable de gérer les échecs de réseau localement et immédiatement sans attendre la convergence de routage globale [32]. Pour résumé, dans NDN, le plan de routage et le plan d'acheminement sont intelligents et adaptatifs. La principale différence entre eux est : le premier détermine quels chemins sont disponibles pour le plan d'acheminement, tandis que le dernier détermine quels chemins peuvent être utilisés et dans quel ordre.

8.2.2 Routage avec Qualité de Service

L'Internet se caractérise par l'hétérogénéité de ses liens et les conditions de trafic dynamiques. Cela nécessite de prendre en compte la qualité de service (QdS) dans le réseau et surtout dans le routage pour répondre aux exigences des utilisateurs. Le but du routage QdS est de trouver un chemin réseau qui satisfasse les contraintes données et optimise simultanément l'utilisation des ressources [50]. L'intégration des paramètres QdS augmente la complexité des algorithmes de routage. En d'autres termes, le problème de la détermination du chemin qui satisfait les contraintes de QdS (deux paramètres de QdS ou plus non corrélés sur un chemin, par exemple, le délai et le coût) est connu pour être NP-complet [51]. Le problème de routage basé sur la QdS a été largement discuté dans la littérature pour l'architecture TCP/IP ; alors qu'il vient de commencer dans les architectures ICN.

8.2.3 Formulation du Problème

Nous formulons le routage avec QdS dans le réseau NDN comme suit. Soit $G = (X, U)$ le graphe constitué d'un ensemble X avec $|X| = N$ nœuds et un ensemble U avec $|U| = M$ liens. Les nœuds représentent les routeurs, les consommateurs ou les fournisseurs, tandis que les liens représentent les canaux de communication. Un lien spécifique de l'ensemble U entre les nœuds u et v est noté (u, v). Chaque lien (u, v) de U est caractérisé par un vecteur m-dimensionnel W tel que $W(u, v) = [w_1(u, v),\ w_2(u, v),\ ...\ w_m(u, v)]$. Si (u, v) appartient à U et m réfère les composants renvoient aux critères QdS tels que $delai$ et $coût$ fournis par le lien (u, v). Enfin, soit $L(u, v) = [L_1(u, v),\ L_2(u, v),\ ...,\ L_m(u, v)]$. Où $L_i(u, v) \geq 0$, est le vecteur des contraintes QdS définies par l'utilisateur (ou l'application).

Le consommateur exprime les données désirées en envoyant au réseau un paquet d'intérêt qui contient le nom de la donnée ainsi que les contraintes de qualité de service. Si la donnée demandée existe, elle prend le même chemin parcouru par l'Intérêt, mais dans la direction opposée. En effet, la donnée peut être trouvée au niveau du nœud fournisseur, d'un dépôt ou d'un cache d'un routeur dans le réseau.

Un chemin dans la topologie G est noté $P(Cons, X)$. Il connecte le nœud consommateur (noté $Cons$), qui a demandé la donnée et X, qui est le nœud qui a une copie de la donnée demandée par le consommateur.

Les algorithmes de routage avec QdS permettent de calculer le chemin P qui optimise un ou plusieurs critères de QdS. L'idée principale de notre travail est de collecter en temps réel différents paramètres dynamiques de QdS et de les exploiter pour guider les différentes demandes de données utilisateur vers des sources potentielles de données via le meilleur chemin P et pour rediriger les données trouvées vers les consommateurs avec la qualité de service requise. Pour atteindre cet objectif, nous procédons par étapes. Tout d'abord, nous surveillons les liens de réseau pour estimer

les paramètres QdS. Ensuite, ces estimations sont collectées par les routeurs. Enfin, les routeurs exploitent ces informations collectées pour déterminer les chemins qui permettent de satisfaire les contraintes de qualité de service requises par l'utilisateur (ou l'application).

Les trois tâches mentionnées précédemment appartiennent au plan d'acheminement. Ce dernier, qui est un trafic bidirectionnel, permet d'observer les performances d'acheminement et d'estimer les paramètres QdS des liens, de les collecter et enregistrer dans la table FIB des routeurs. Ces informations appelées information du plan d'acheminement, ainsi que les informations du plan de routage (enregistrées aussi dans la table FIB) sont utilisées comme entrées pour notre stratégie d'acheminement, qui détermine les chemins permettant de satisfaire les contraintes de qualité du service.

Sur la base des informations présentées dans cette formulation, la solution que nous proposons pour résoudre le problème de routage basé sur la QdS, tout au long de cette thèse, consiste en la conception et la mise en œuvre de nouvelles stratégies d'acheminement qui permettent de collecter en temps réel des mesures de paramètres de QdS des liens du réseau et les intégrer dans les différentes décisions prises pour transmettre les requêtes des utilisateurs.

8.3 Nos contributions

8.3.1 QoS-FS : Stratégie d'acheminement basée sur la QoS pour NDN

La première contribution consiste en la conception et la mise en œuvre d'une nouvelle stratégie d'acheminement adaptative avec qualité de service (QdS) pour NDN, qui ne considèrent que deux paramètres de QdS (Round Trip Time (RTT) et la bande passante), Cette stratégie est appelée **QoS-FS**. Au niveau de chaque nœud du réseau, **QoS-FS** surveille, en temps réel, les liaisons entrantes et sortantes du réseau pour estimer les paramètres de QdS, et les intégrer dans les différentes décisions prises pour déterminer quand et quelle interface utiliser pour transmettre un paquet entrant. Par conséquent, rendre la décision d'expédition adaptative aux conditions du réseau et les préférences de l'utilisateur. La conception de **QoS-FS** est basée sur une méthode d'apprentissage inductive, le type Q-learning en ligne, que nous avons utilisé pour estimer les informations collectées sur l'état dynamique du réseau.

Nous avons évaluer la stratégie d'acheminement proposée **QoS-FS** sous le simulateur ndnSIM 2.0 [71],et nous avons comparé ses performances avec la Stratégie d'acheminement Best Route [70]. Notre analyse est basée sur deux métriques : *data delivery time* qui représente le temps écoulé entre l'envoi d'intérêt et la réception de données, et $hopscount$ qui représente le nombre de sauts parcourus par les données pour arriver au consommateur. La topologie $Abilene$ est utilisée comme noyau du réseau simulé, qui est composé de 12 routeurs. Dans le scénario de simulation, chaque routeur principal est rattaché à un nombre variable entre 2 et 4 des consommateurs et 1 serveur pour stocker des contenus permanents. Par consequent,le nombre total de nœuds égal à

64. Nous utilisons un catalogue de 100 contenus, chaque contenu peut avoir entre 1 et 4 répliques au hasard dans les différents référentiels. Les consommateurs demandent un contenu selon la loi de Zipf (voir $ns3 :: ndn :: ConsumerZipfMandelbort$ [34]) pour d'autres détails de simulation, nous avons les valeurs par défaut pour les paramètres $q = 0,7$ et $s = 0,7$ Qui représentent respectivement le paramètre d'amélioration du rang et du paramètre de puissance.

Plusieurs simulations ont été effectué pour chaque fréquence de demande de consommateur suivante : 10, 20, 30, 50, 70 et 100 Intérêts par seconde. Ensuite, nous calculons la moyenne du temps de livraison des données et la moyenne du nombre de sauts pour chaque simulation. Les figures 8.1 et 8.2 résument le temps moyen de livraison des données et les résultats moyens du compte-sauts des simulations, respectivement.

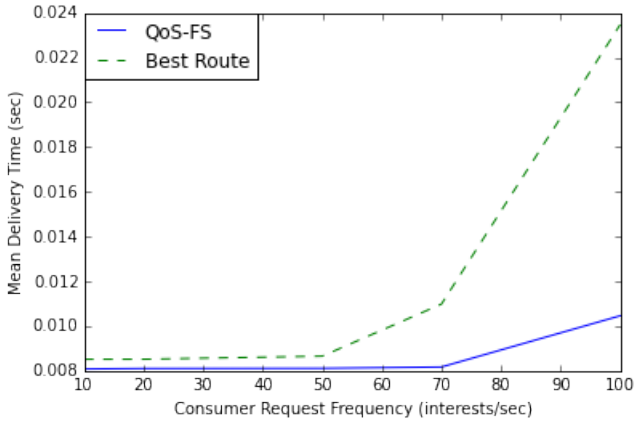


FIGURE 8.1 : Délai moyen de livraison.

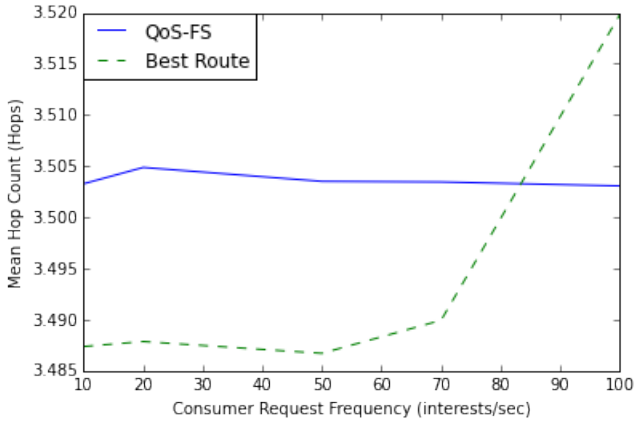


FIGURE 8.2 : Nombre moyen de sauts.

La figure 8.1 montre que lorsque la fréquence de demande des consommateurs est inférieure à

50 pour la stratégie d'acheminement Best route, le temps de livraison moyen est presque le même. Entre 50 et 70, il augmente linéairement, mais au-delà de 70, il augmente rapidement. Pour **QoS-FS**, le délai moyen de livraison est presque identique jusqu'à la fréquence 70, mais après cela, il commence à augmenter linéairement. Les résultats obtenus montrent clairement que le **QoS-FS** peut traiter efficacement la fréquence de demande du consommateur par rapport à la stratégie Best route. En effet, **QoS-FS** donne toujours un meilleur délai de livraison que Best Route.

La figure 8.2 montre la comparaison entre **QoS-FS** et Best Route en termes de nombre de saurts moyen. Pour la meilleure route, il est stable lorsque la fréquence de la demande du consommateur est inférieure à 50, et elle commence à augmenter linéairement entre 50 et 70, mais au-delà, elle augmente rapidement en dépassant le nombre de hops moyen **QoS-FS** qui est toujours toujours stable. Cela s'explique par le fait que **QoS-FS** répartit judicieusement le trafic sur tous les liens en fonction de l'état du réseau.

Les figures 8.1 et 8.2 montrent que malgré le nombre de sauts moyen de Best Route soit inférieur à celui de **QoS-FS**, ce dernier donne un meilleur délai de livraison des données. Ainsi, le meilleur chemin n'est pas nécessairement celui qui a le plus petit nombre de sauts.

Les résultats obtenus expliquent l'effet positif de l'intégration de la bande passante des liens dans les décisions, ce qui est derrière la bonne distribution du flux sur les chemins disponibles. Cela permet d'offrir un bon délai de livraison avec un nombre moyen de sauts constant, dans tous les scénarios d'expérimentation.

8.3.2 AC-QoS-FS : Stratégie d'acheminement avec qualité de service basée sur les colonies de fourmis

La deuxième contribution consiste également en la conception et la mise en œuvre d'une autre stratégie d'acheminement adaptative avec QdS pour le réseau NDN, appelée : **AC-QoS-FS**. Elle est basée sur les similitudes entre le processus d'acheminement du NDN et le comportement des fourmis tout en recherchant le chemin le plus court entre leur nid et une source de nourriture. Les techniques utilisées pour concevoir **AC-QoS-FS** sont empruntées de l'algorithme d'optimisation des colonies de fourmis (Ant System) qui a été à l'origine utilisé pour résoudre le fameux problème NP-Complet du voyageur de commerce (PVC). La stratégie d'acheminement **AC-QoS-FS** est capable de prendre en compte trois paramètres QdS (le temps d'aller-retour (RTT), la bande passante et le coût). En d'autres termes, nous avons adapté cet algorithme pour concevoir **AC-QoS-FS**. Par conséquent, au départ, les fourmis explorent des chemins plus ou moins aléatoires. Ensuite, au fil du temps, ils seront adaptés à l'environnement et les chemins optimaux émergeront, qui seront les plus suivis. Donc, au début, à chaque nœud, les fourmis sont envoyées aléatoirement au prochain saut en amont via les interfaces disponibles. Lorsque la fourmi trouve la donnée demandée, elle retourne au nœud consommateur en suivant le même chemin parcouru pendant la phase de recherche, mais dans le sens inverse. En outre, au niveau de chaque nœud, elle dépose une quantité de phéromone en

fonction de la qualité, en temps réel, du chemin parcouru entre la source alimentaire (fournisseur ou mémoire cache d'un routeur) et le nœud courant, en termes de mesures des paramètres de QdS collectés par la fourmi. Ces mesures de paramètres QdS sont mises à jour au niveau de chaque nœud avant d'acheminer le paquet vers le prochain saut en aval. Ensuite, au niveau de chaque nœud, **AC-QoS-FS** classe les interfaces disponibles et prend les décisions d'acheminement des intérêts entrants en fonction de la quantité de la phéromone.

Nous avons évaluer la stratégie d'acheminement proposée **AC-QoS-FS** également sous le simulateur ndnSIM, et nous avons comparé ses performances aussi avec Best Route en basant sur les métriques suivantes :

- Le temps moyen de livraison des données, qui représente le temps moyen écoulé entre l'envoi d'intérêt et la réception des données.
- Le coût moyen, qui représente la moyenne de la somme des coûts des liens parcourue par les paquets de données : de la source de données jusqu'au consommateur.
- Le nombre moyen de sauts, représentant le nombre moyen de sauts parcourus par les données entre la source de données et le consommateur.
- La quantité des paquets supprimés en Kilo octet.
- Le taux de succès, qui représente le rapport du nombre total de cache sur le nombre succès total de cache (manqué et cache).

Nous avons utilisé la topologie d'Abilene comme noyau du réseau simulé, composé de 12 routeurs. Dans le scénario de simulation, chaque routeur de base est attaché à 1 serveur de stockage utilisé pour stocker des contenus permanents ainsi un nombre variable entre 2 et 4 de consommateurs. Par consquence, le nombre total de nœuds est égal à 64. Nous avons utilisé un catalogue de 100000 contenus, où chaque contenu possède 3 répliques dans les différents serveurs et chaque routeur peut stocker jusqu'à 1000 morceaux dans son CS.

Les consommateurs demandent les contenus selon la loi de Zipf largement adoptée, et en ce qui concerne la popularité des contenus, nous avons considré les valeurs de paramètre de popularité $\alpha = 0.8; 1.0; 1.2$ [103] avec différentes fréquences : 10, 20 et 30 Intérêts par seconde.

Il convient de mentionner que les résultats **AC-QoS-FS** sont représentés dans les figures 8.3(a) , 8.3(b) , 8.3(c) , 8.3(d) et 8.3(e) , ont été obtenus avec la configuration suivante : $\alpha = \beta = \gamma = 1/3$ ce qui signifie que RTT, bande passante et coût ont la même importance pour l'application ou la fin Utilisateur et nous avons réglé $\rho = 0.2$ et $Q = 100$. Figure 8.3(a) compare le coût moyen associé aux transferts de données en utilisant notre stratégie **AC-QoS-FS** et Best Route. Pour une fréquence de 10 requêtes par seconde, notre stratégie est approximativement de $8\% - 10\%$ plus élevé que Best Route, pour les trois valeurs du paramètre Zipf-law α. Cependant, il existe une

tendance intéressante montrant que, plus la fréquence des demandes est élevée, le coût de **AC-QoS-FS** moins. Dans le cas d'une fréquence égale à 30, la réduction des coûts reste autour de 10% pour les 3 valeurs de α. Cela montre que, pour les scénarios avec des charges croissantes (plusieurs demandes qui circulent), notre stratégie a tendance à coûter moins cher.

Figure 8.3(b) montre que les tau de succès pour notre solution est, en général, moins important que l'autre stratégie. En conséquence, le nombre moyen de sauts qu'une fourmi doit effectuer est légèrement plus élevé pour **AC-QoS-FS**, comme on le montre la Figure 8.3(c) . Ce phénomène s'explique en raison de la nature même de nos algorithmes, qui repose sur une sélection aléatoire du chemin à suivre par la fourmi, en fonction des quantités de phéromone à chaque routeur. Cela implique que, un chemin menant à un prochain saut avec un faible taux de probabilité de succès peut également être sélectionné. Cependant, la figure 8.3(d) montre que ce comportement n'a pas d'impact significatif sur le temps moyen de chemin, car il reste à peu près identique pour les deux solutions. De même, le taux de paquets supprimés est similaire, même si les fourmis effectuent des sauts plus longs pour récupérer l'information, comme le montre la Figure 8.3(e) .

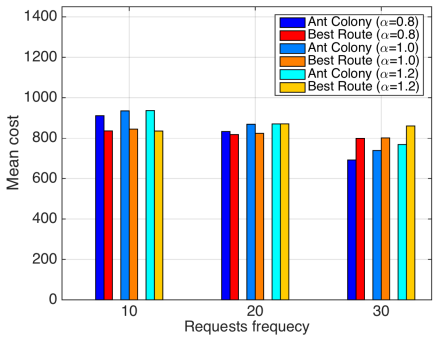
Les résultats de différents scénarios que nous avons exécutés démontrent que nous pouvons concevoir une stratégie d'acheminement qui pourrait assurer les meilleures performances en réglant ($\alpha, \beta, \gamma, \rho$ et Q) selon l'application ou la fin Exigences des utilisateurs.

8.3.3 MC-QoS-FS : Stratégie d'acheminement basée sur multi critères de qualité de service

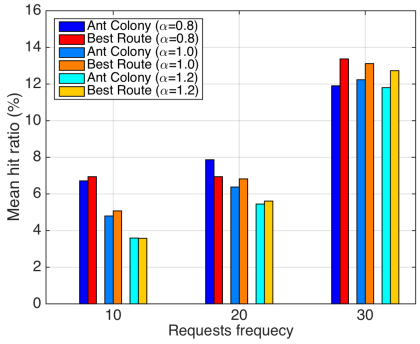
Dans la troisième contribution, au début nous avons généralisé l'approche utilisée dans la deuxième contribution pour supporter plusieurs paramètres de QdS, en proposant un modèle pour concevoir des stratégies d'acheminements basées sur multi critères de QdS. Ensuite, en appliquant les principes proposés dans ce modèle, nous avons conçu et mis en œuvre une nouvelle stratégie d'acheminement appelée (**FCR-QoS-FS**) : stratégie d'acheminement avec qualité de service basée sur les colonies de fourmis pour la réduction du flux de congestion. **FCR-QoS-FS** prend en compte cinq paramètres QdS, elle est conçue pour s'adapter, autant que possible, au trafic dynamique du réseau, en particulier aux problèmes de congestion. Par conséquent, en plus d'optimiser les paramètres QdS pris en compte dans la deuxième contribution (le temps d'aller-retour (RTT), la bande passante et le coût), elle prend également en compte deux autres paramètres qui ont un impact très élevé sur la congestion qui sont taux d'intérêt arrivés (IAR) et taux de données satisfaites (DSR).

Afin d'évaluer la stratégie d'acheminement proposée **FCR-QoS-FS**, le simulateur ndnSIM a été utilisé. Nous avons comparé les performances de la stratégie d'acheminement proposée **FCR-QoS-FS** avec **AC-QoS-FS** et Best Route. L'analyse des performances des stratégies a été effectué sur la base de même métriques cité précédement.

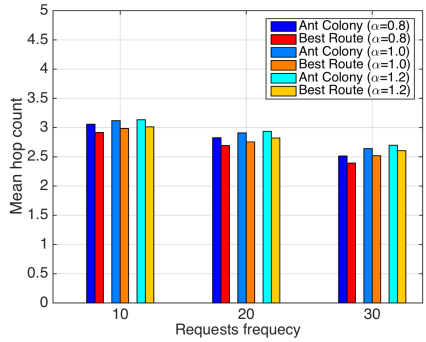
Nous avons utilisé la topologie d'Abilene composée de 12 routeurs comme noyau du réseau simulé pour évaluer les performances des stratégies. Dans les scénarios de simulation, chaque rou-



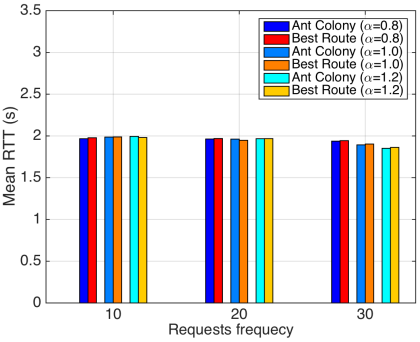
((a)) Coût moyen.



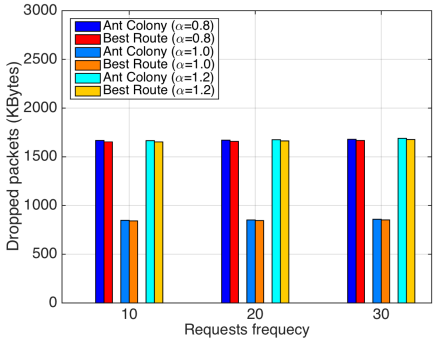
((b)) Taux de réussite.



((c)) Nombre moyen de sauts.



((d)) Délai moyen de livraison.



((e)) Quantité de paquets supprimés en kilo octet.

FIGURE 8.3 : Résultats obtenus.

teur de noyau est attaché à un serveur de stockage utilisé pour stockage permanent des contenus et il est également attaché à un nombre variable entre 2 et 4 de consommateurs. Par consequence, le nombre total de nœuds est égal à 64. Nous avons utilisé un catalogue de 100000 Contenus, où chaque contenu possède 3 répliques aléatoires dans les différents serveurs et chaque routeur peut stocker jusqu'à 1000 morceaux dans son CS. Pour remplir le tableau FIB des nœuds, nous avons

utilisé la bibliothèque ***ndnGlobalRoutingHelper*** fournie par le simulateur ndnSIM.

Dans les scénarios effectués, les consommateurs demandent des contenus selon la loi de Zipf [103]. Les valeurs de paramètre de popularité de contenu considéré sont($\alpha = .8; 1.0;$ et 1.2) avec différentes fréquences (3, 5, 7, 10, 15, 20 et 30 intérêts par seconde).

Il convient de mentionner que les résultats de la stratégie d'acheminement proposée **FCR-QoS-FS** ont été obtenus avec la configuration suivante : $\alpha = \beta = \gamma = \theta = \mu = 1/5$ qui signifie que RTT, bande passante, coût, IAR et DSR ont la même importance pour l'application ou l'utilisateur final et nous avons également reglé $\rho = 0,2$ et $Q = 100$. Tandis que, les résultats de la stratégie d'acheminement **AC-QoS-FS** ont été obtenus avec la configuration suivante : $\alpha = \beta = \gamma = 1/3$. Les paramètres RTT, la bande passante et le coût ont également la même importance. En outre, nous avons défini $\rho = 0,2$ et $Q = 100$. Ces résultats sont représentés dans les Tableaux 6.3, 6.4 et Figure 8.4. Ce dernier représente la quantité des paquets supprimés pour chaque stratégie.

En analysant les résultats obtenus compte tenu de toutes les fréquences et des valeurs de popularité paramètre α mentionné ci-dessus, nous pouvons constater que la Stratégie d'acheminement Best Route offre une bonne performance relative par rapport à **AC-QoS-FS** en ce qui concerne Le nombre moyen de sauts, le coût et le taux de succès Tables 6.3 et 6.4.

Comme le principe utilisé par l'approche Best Route repose sur la réduction des coûts , elle dirige les Intérêts entrants vers les chemins de coût inférieur, ce qui explique les bons résultats obtenus en termes de RTT, de coût et de nombre de sauts par rapport à nos stratégies d'acheminement proposées, mais elle fournit les pires performances concernant la quantité des paquets supprimés.

Sachant que les résultats obtenus pour le coût, le nombre de sauts et le RTT ne sont mesurés que pour les paquets reçus par le consommateur, et les paquets supprimés ne sont pas pris en compte. Si nous considérons la quantité importante des paquets supprimés pour Best Route, nous pouvons finalement conclure que nos propositions (en particulier **FCR-QoS-FS**) offrent les meilleures performances pour tous les métriques de comparaison.

Les résultats obtenus montrent que **AC-QoS-FS** fournit une meilleure performance par rapport **FCR-QoS-FS**, pour le coût, le nombre de sauts et le tau de succès. En ce qui concerne RTT, les résultats obtenus sont comparables à une préférence légèrement meilleure pour **FCR-QoS-FS**. En termes de quantité de paquets supprimés, **FCR-QoS-FS** fournit les meilleures performances. Comme mentionné précédement, puisque le coût, le nombre de sauts et le RTT ne sont mesurés que pour les paquets reçus par les consommateurs, et en vue la différence expressive concernant la quantité de paquets supprimés, il est clair que le **FCR-QoS-FS** est plus efficace que le **AC-QoS-FS**.

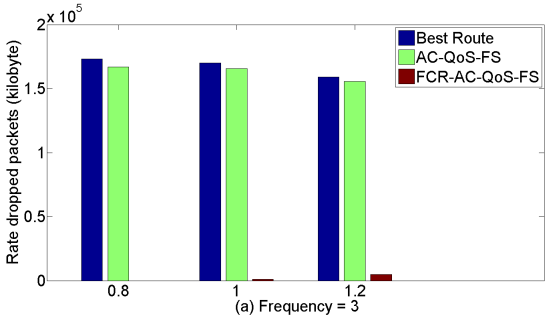
Etant donné que **AC-QoS-FS** et **FCR-QoS-FS** sont conçues suivant le même modèle sauf que **FCR-QoS-FS** prend en considération les paramères IAR et DSR en plus aux paramètres prise par **AC-QoS-FS**, il est évident que l'addition de ces deux paramètres conduit à minimiser la quantité des paquets supprimés, parce que **FCR-QoS-FS** sur la base des valeurs de ces deux paramètres, elle dirige les paquets vers les chemins moins congestionnés.

La figure 8.4 démontre que **FCR-QoS-FS** est plus efficace que **AC-QoS-FS** et Best Route en ce qui concerne la quantité de paquets supprimés en kilo octet et Table 8.1 démontre en détails les taux (en %) de la quantité de paquets supprimés pour **FCR-QoS-FS** contre **AC-QoS-FS** et Best Route. Par exemple, lorsque la fréquence est égale 3 requêtes par seconde et le paramètre de popularité $\alpha = 0.8$ le **FCR-QoS-FS** ne supprime aucun paquet, alors que **AC-QoS-FS** et Best Route suppriment respectivement 167005 et 173120 Kilo octet, donc **FCR-QoS-FS** supprime (-100 %) par rapport **AC-QoS-FS** et **Best Route**. Cependant, lorsque la fréquence est égale 15 requêtes par seconde et le paramètre de popularité $\alpha = 1.2$ the **FCR-QoS-FS** supprime 1239498 Kilo octet, alors que **AC-QoS-FS** et Best Route suppriment respectivement 1754740 et 1755011 Kilo octet, donc **FCR-QoS-FS** supprime (-29,362 %) par rapport **AC-QoS-FS** et (-29,373 %) par rapport **Best Route**.

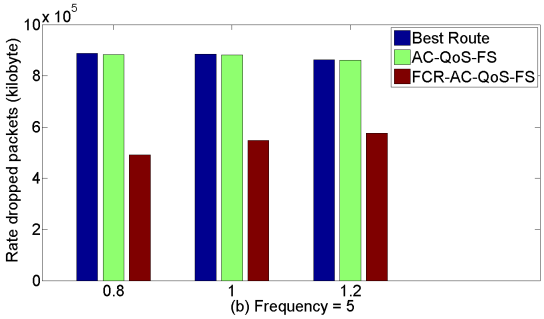
Fréquence	alpha	AC-QoS-FS	Best Route
3	0.8	-100 %	-100 %
	1.0	-99.3618048015442 %	-99.3781993640942 %
	1.2	-96.8346822146939 %	-96.9040014084153 %
5	0.8	-44.4031514454687 %	-44.6907581511124 %
	1.0	-37.8489787633119 %	-38.0137462719147 %
	1.2	-33.0863415516977 %	-33.1874163071833 %
7	0.8	-45.2181345178457 %	-45.3636464886612 %
	1.0	-39.8681350565153 %	-39.9653265293254 %
	1.2	-34.592659375195 %	-34.6046508667061 %
10	0.8	-44.5370841781781 %	-44.6691997354849 %
	1.0	-41.4195315228575 %	-41.5306389831856 %
	1.2	-32.7822556783927 %	-32,8089860002249 %
15	0.8	-46.2533838626038 %	-46.4485090885255 %
	1.0	-40.7462872831649 %	-40.8931009557465 %
	1.2	-29.3628685731219 %	-29.3737760048228 %
20	0.8	-45.4473794793911 %	-45.6976243516936 %
	1.0	-39.227870163515 %	-39.4050737230693 %
	1.2	-25.94385050228 %	-25.9722370274962 %
30	0.8	-48.2782015329356 %	-48.6860309526296 %
	1.0	-39.6771460665034 %	-40.0195713149975 %
	1.2	-25.2228852979962 %	-25.3386377997717 %

TABLE 8.1 : Le taux (en %) de la quantité des paquets supprimés.

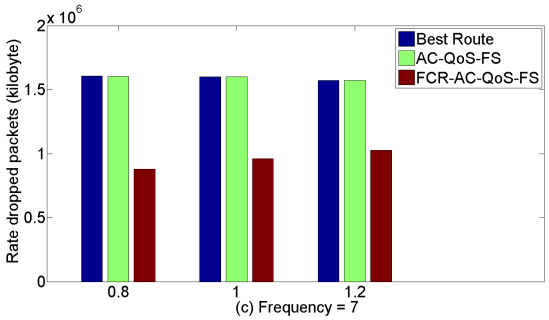
Nous pouvons notamment remarquer que, lorsque la fréquence augmente, la quantité de paquets supprimés augmente également pour toutes les stratégies. En outre, les valeurs du RTT, du coût et du nombre de sauts diminuent avec la valeur croissante du paramètre α pour toutes les fréquences, et les valeurs de ces mesures diminuent également avec l'augmentation des fréquences utilisées. Enfin, nous pouvons remarquer que RTT du *FCR-AC-QoS-FS* devient meilleur par rap-



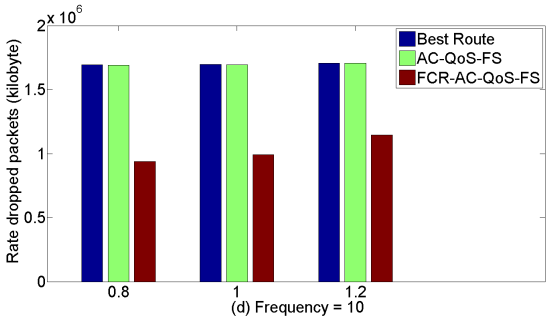
((a)) Frequency=3



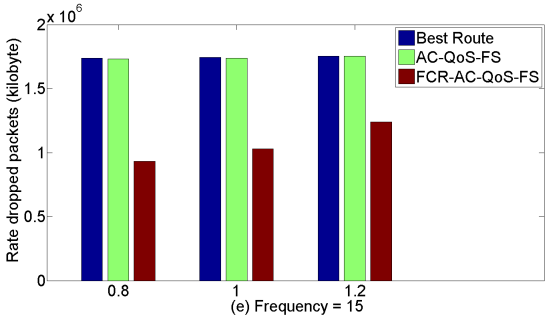
((b)) Frequency=5



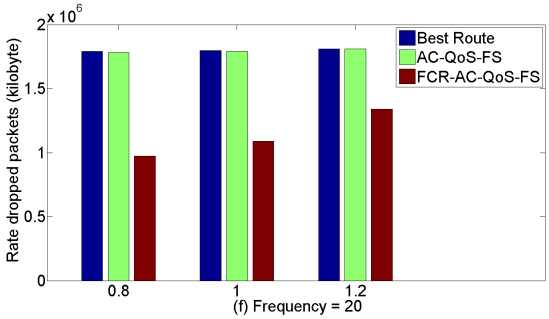
((c)) Frequency=7



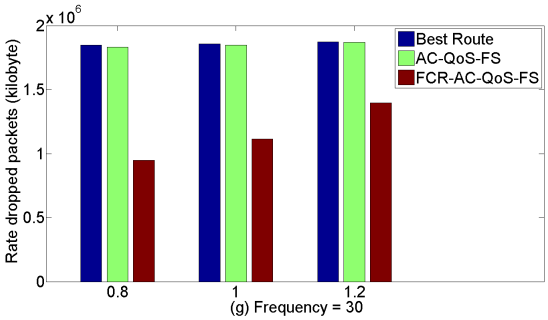
((d)) Frequency=10



((e)) Frequency=15



((f)) Frequency=20



((g)) Frequency=30

FIGURE 8.4 : Quantité des pacquets supprimés en kilo octet.

port aux autres stratégies pour les fréquences 20 et 30. Nous soulignons que, malgré les résultats obtenus montrent que les stratégies d'acheminement proposée donnent très bonnes performances, mais il est possible de les améliorer, par réglage précis de ($\alpha, \beta, \gamma, \theta, \mu, \rho$ et Q) selon exigences de l'application ou des utilisateurs finaux.

8.4 Conclusion

L'objectif de cette thèse était d'étudier la problématique du routage dans les réseaux conçus selon le paradigme Information Centric-Networking (ICN). Nos contributions étaient selon l'architecture du réseau NDN (Named Data Networking) qui est considéré actuellement comme la plus avancée. Nous avons étudié le routage avec QdS qui vient de démarrer dans les architectures ICN et en NDN en particulier, bien que, il a été largement discuté dans la littérature pour l'architecture IP. Étant donné que le plan d'acheminement à l'aide de son module stratégie d'acheminement est responsable du processus d'acheminement des paquets et que le routage est juste responsable de la construction de la table FIB et pour aider le plan d'acheminement à prendre les meilleures décisions, par les informations qu'il offre. Nos contributions dans cette thèse consistent en la conception et à la mise en œuvre de trois nouvelles stratégies d'acheminement adaptatives avec QdS. L'analyse des performences des stratégies proposées montrent l'importance de la prise en considération de l'état du réseau en temps réel en matière des paramètres de QdS. Ainsi l'importance de choix des paramètres et leurs poids qui est liée aux exigences de l'application ou l'utilisateur final.

De ce travail, de nombreuses perspectives peuvent être formulées, que nous proposons comme suit :

- Exploration de notre dernière proposition en utilisant différents types et nombre de paramètres QdS.

- Afin de confirmer l'efficacité de nos stratégies d'acheminement proposées, nous devrions étudier leur évaluation des performances en les testant dans le scénario naturel lorsqu'ils travaillent simultanément avec les protocoles de routage disponibles tels que NLSR en mode simulation sous le simulateur ndnSIM. Malheureusement, jusqu'à présent, la version NLSR sur ndnSIM n'existe pas, mais elle sera disponible dans le futur.

- Extension de nos stratégies d'acheminement proposées pour qu'elles puissent prendre en charge différentes pannes de réseau telles que : l'échec de la liaison, le détournement des préfixes, etc. En outre, nous pouvons étudier l'adaptation des stratégies d'acheminements proposées pour soutenir différentes politiques de mise en cache afin d'atteindre la meilleure efficacité.

- Etudier d'autres algorithmes d'optimisation pour les utiliser dans la gestion de l'acheminement intelligent des paquets sur multipaths sur la base de la qualité de service, tout en considérant les paramètres QoS visant à atteindre plus d'efficacité.
- Etudier d'autres algorithmes d'optimisation pour les utiliser dans gestion du routage basé sur la QdS et l'acheminement intelligent des paquets sur les chemins multiples, tout en considérant les paramètres QdS en visant à atteindre plus d'efficacité.
- Evaluer les performances de nos propositions sur un véritable testbed. Par exemple, sur une plateforme NDN basée sur les routeurs Banana Pi.
- Il est à noter que nos contributions de thèse sont considérées parmi les premiers travaux dans la littérature qui traitent la problèmatique de routage avec QdS dans les réseaux conçue selon le paradigme ICN, et dans les réseaux NDN en particulier. Cependant, selon nos connaissances jusqu'à maintenant, il n'y a pas de travaux de recherche dans la littérature qui traitent la problèmematique de routage avec qualité d'expérience (QdE). Par conséquent, cette problèmatique pourrait être étudiée.

Bibliographie

[1] M. Handley, "Why the internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, 2006.

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

[3] Statistic, "Internet of things (iot) : number of connected devices worldwide from 2012 to 2020 (in billions)."

[4] J. Rexford and C. Dovrolis, "Future internet architecture : clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, 2010.

[5] A. Taran and D. Lavrov, "Future internet architecture : Clean-slate vs evolutionary design," , no. 3 (39), 2016.

[6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, ACM, 2009.

[8] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[9] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[10] N. D. nNtworking Team, "Named data networking project."

[11] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 44–53, 2012.

[12] V. Jacobson, "A new way to look at networking," *Google Tech Talk*, vol. 30, 2006.

[13] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, Aug. 2007.

[14] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further : From psirp to pursuit," in *International Conference on Broadband Communications, Networks and Systems*, pp. 1–13, Springer, 2010.

[15] D. Lagutin, K. Visala, and S. Tarkoma, "Publish/subscribe for internet : Psirp perspective.," *Future internet assembly*, vol. 84, 2010.

[16] C. Dannewitz, J. Golic, B. Ohlman, and B. Ahlgren, "Secure naming for a network of information," in *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pp. 1–6, IEEE, 2010.

[17] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, and P. Hallam-Baker, "Naming things with hashes," tech. rep., 2013.

[18] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[19] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht : a hierarchical name resolution service for information-centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 7–12, ACM, 2011.

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

[21] M. F. P. Team, "Nsf mobility first project."

[22] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Comput Commun*, vol. 36, pp. 779–791, Apr. 2013.

[23] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM computer communication review*, vol. 42, no. 3, pp. 62–67, 2012.

[24] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, pp. 779–791, apr 2013.

[25] V. Jacobson, J. Burke, D. Estrin, L. Zhang, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, *et al.*, "Named data networking (ndn) project 2012-2013 annual report," 2014.

[26] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data," in *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pp. 3–8, ACM, 2010.

[27] C. Team, "The convergence project."

[28] S. Team, "Scalable and adaptive internet solutions (sail)."

[29] C. Team, "Content mediator architecture for content-aware networks (comet)."

[30] T. F. W. Team, "The fp7 4ward project."

[31] A. connect Team, "Reseaux du futur et services (verso) 2010 projet connect."

[32] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, (New York, NY, USA), pp. 27–36, ACM, 2014.

[33] V. Jacobson, J. Burke, L. Zhang, B. Zhang, K. Claffy, C. Papadopoulos, T. Abdelzaher, L. Wang, J. A. Halderman, and P. Crowley, "Named data networking next phase (ndn-np) project may 2015-april 2016 annual report,"

[34] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "Ns-3 based named data networking (ndn) simulator ndnsim : simulation of ndn, icn, information-centric networking."

[35] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "Ospfn : An ospf based routing protocol for named data networking," *University of Memphis and University of Arizona, Tech. Rep*, 2012.

[36] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "Nlsr : named-data link state routing protocol," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pp. 15–20, ACM, 2013.

[37] V. Lehman, A. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, "A secure link state routing protocol for ndn," *NDN, Technical Report NDN-0037*, 2016.

[38] V. Lehman, A. M. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, "A secure link state routing protocol for ndn,"

[39] N. P. Team, "The ndn platform."

[40] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto, *et al.*, "Nfd developer's guide," *Technical Report NDN-0021, NDN*, 2014.

[41] Z. Zhu and A. Afanasyev, "Let's chronosync : Decentralized dataset state synchronization in named data networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2013.

[42] N. P. Team, "Nlsr 0.1.0."

[43] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.

[44] M. Boguná, F. Papadopoulos, and D. Krioukov, "Sustaining the internet with hyperbolic mapping," *Nature communications*, vol. 1, p. 62, 2010.

[45] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. Krioukov, and L. Wang, "An experimental investigation of hyperbolic routing with a smart forwarding plane in ndn," in *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, pp. 1–10, IEEE, 2016.

[46] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, E. Uzun, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, *et al.*, "Named data networking (ndn) project 2011-2012 annual report,"

[47] N. Team *et al.*, "Named data networking (ndn) project 2012-2013 annual report," 2013.

[48] V. Jacobson, J. Burke, L. Zhang, B. Zhang, K. Claffy, D. Krioukov, C. Papadopoulos, T. Abdelzaher, L. Wang, E. Yeh, *et al.*, "Named data networking (ndn) project 2013-2014 report," 2014.

[49] V. Jacobson, J. Burke, L. Zhang, B. Zhang, K. Claffy, C. Papadopoulos, T. Abdelzaher, L. Wang, J. A. Halderman, and P. Crowley, "Named data networking next phase (ndn-np) project may 2014-april 2015 annual report,"

[50] M. R. Gary and D. S. Johnson, "Computers and intractability : A guide to the theory of np-completeness," 1979.

[51] F. A. Kuipers and P. F. Van Mieghem, "Conditions that impact the complexity of qos routing," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 4, pp. 717–730, 2005.

[52] A. Udugama, X. Zhang, K. Kuladinithi, and C. Goerg, "An on-demand multi-path interest forwarding strategy for content retrievals in ccn," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–6, IEEE, 2014.

[53] D. Posch, B. Rainer, and H. Hellwagner, "Saf : Stochastic adaptive forwarding in named data networking," *arXiv preprint arXiv :1505.05259*, 2015.

[54] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "Inform : A dynamic interest forwarding mechanism for information centric networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, (New York, NY, USA), pp. 9–14, ACM, 2013.

[55] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown : Hamlet-like doubts in icn," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN 12, (New York, NY, USA), pp. 7–12, ACM, 2012.

[56] G. Rossini and D. Rossi, "Coupling caching and forwarding : Benefits, analysis, and implementation," in *Proceedings of the 1st international conference on Information-centric networking*, pp. 127–136, ACM, 2014.

[57] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[58] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim : An highly scalable ccn simulator," in *Communications (ICC), 2013 IEEE International Conference on*, pp. 2309–2314, IEEE, 2013.

[59] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp : Design and evaluation of an interest control protocol for content-centric networking," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 304–309, IEEE, 2012.

[60] L. Saino, C. Cocora, and G. Pavlou, "Cctcp : A scalable receiver-driven congestion control protocol for content centric networking," in *Communications (ICC), 2013 IEEE International Conference on*, pp. 3775–3780, IEEE, 2013.

[61] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pp. 363–368, IEEE, 2013.

[62] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, "An empirical study of receiver-based aimd flow-control strategies for ccn," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pp. 1–8, IEEE, 2013.

[63] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pp. 37–42, ACM, 2012.

[64] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 322–327, IEEE, 2012.

[65] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 55–60, ACM, 2013.

[66] Y. Wang, "Caching,routing and congestion control in a future information-centric internet," *http ://www.lib.ncsu.edu/resolver/1840.16/9009*, 2013.

[67] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in information-centric networks," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pp. 1–10, IEEE, 2013.

[68] G. Carofiglio, M. Gallo, and L. Muscariello, "Optimal multipath congestion control and request forwarding in information-centric networks : Protocol design and experimentation," *Computer Networks*, vol. 110, pp. 104–117, 2016.

[69] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking*, Association for Computing Machinery (ACM), 2016.

[70] I. M. Spyridon Mastorakis, Alexander Afanasyev and L. A. Lixia Zhang University of California, "ndnsim 2.0 : A new version of the ndn simulator for ns-3," in *NDN, Technical Report NDN-0028, 2015, Revision 1 : January 27, 2015*, 2015.

[71] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim : Ndn simulator for ns-3," in *NDN, Technical Report NDN-0005, 2012.Revision 2 : October 5, 2012*, 2012.

[72] L. Chengming, L. Wenjing, and K. OKAMURA, "A greedy ant colony forwarding algorithm for named data networking," *Proceedings of the Asia-Pacific advanced network*, vol. 34, pp. 17–26, 2013.

[73] C. Li, K. Okamura, and W. Liu, "Ant colony based forwarding method for content-centric networking," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pp. 306–311, IEEE, 2013.

[74] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system : optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[75] J. Lv, X. Wang, and M. Huang, "Ant colony optimization-inspired icn routing with content concentration and similarity relation," *IEEE Communications Letters*, 2016.

[76] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello, "Soccer : Services over content-centric routing," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 62–67, ACM, 2011.

[77] J. Eymann and A. Timm-Giel, "Multipath transmission in content centric networking using a probabilistic ant-routing mechanism," in *International Conference on Mobile Networks and Management*, pp. 45–56, Springer, 2013.

[78] H. Qian, R. Ravindran, G.-Q. Wang, and D. Medhi, "Probability-based adaptive forwarding strategy in named data networking," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 1094–1101, IEEE, 2013.

[79] K. M. Schneider and U. R. Krieger, "Beyond network selection : Exploiting access network heterogeneity with named data networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*, pp. 137–146, ACM, 2015.

[80] A. Detti, C. Pisa, and N. B. Melazzi, "Modeling multipath forwarding strategies in information centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pp. 324–329, IEEE, 2015.

[81] A. Udugama and C. Goerg, "Adaptation of multi-path forwarding strategies for ccn to operate under mobility," *CCNxCon2013. Parc*, 2013.

[82] G. Rossini and D. Rossi, "Evaluating ccn multi-path interest forwarding strategies," *Computer Communications*, vol. 36, no. 7, pp. 771–778, 2013.

[83] Q. Huang and F. Luo, "Ant-colony optimization based qos routing in named data networking," *Journal of Computational Methods in Sciences and Engineering*, no. Preprint, pp. 1–12, 2016.

[84] A. Bouacherine, M. R. Senouci, and B. Merabti, "Parallel multi-path forwarding strategy for named data networking," in *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications, Volume 1 : DCNET*, pp. 36–46, Scitepress, 2016.

[85] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, "Quality of service extensions to ospf or quality of service path first routing (qospf)," 1997.

[86] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning : A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[87] R. S. Sutton and A. G. Barto, *Reinforcement learning : An introduction*, vol. 1. MIT press Cambridge, 1998.

[88] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the eleventh international conference on machine learning*, vol. 157, pp. 157–163, 1994.

[89] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[90] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[91] R. Bellman and R. Kalaba, "On adaptive control processes," *IRE Transactions on Automatic Control*, vol. 4, no. 2, pp. 1–9, 1959.

[92] A. MELLOUK, *Etude des Systèmes modulaires Neuro-Prédictif : application à la Reconnaissance Automatique de la Palire Continue*. PhD thesis, Thèse de Doctorat Université Paris-Sud, Orsay, 1994.

[93] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks : A reinforcement learning approach," in *In Advances in Neural Information Processing Systems 6, Morgan Kaufmann,*, pp. 671—678, 1994.

[94] Y. Z. Tsypkin, *Foundations of the theory of learning systems*. Academic Press, 1973.

[95] L. Bottou, "Stochastic learning," in *Advanced lectures on machine learning*, pp. 146–168, Springer, 2004.

[96] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[97] J.-L. Deneubourg, J. M. Pasteels, and J.-C. Verhaeghe, "Probabilistic behaviour in ants : a strategy of errors ?," *Journal of Theoretical Biology*, vol. 105, no. 2, pp. 259–271, 1983.

[98] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[99] R. Beckers, J.-L. Deneubourg, and S. Goss, "Trails and u-turns in the selection of a path by the ant lasius niger," *Journal of theoretical biology*, vol. 159, no. 4, pp. 397–415, 1992.

[100] N. Monmarché, *Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation*. PhD thesis, Université François Rabelais-Tours, 2000.

[101] M. Dorigo and L. M. Gambardella, "Ant colony system : a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[102] L. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proceedings of IEEE International Conference on Evolutionary Computation*, Institute of Electrical and Electronics Engineers (IEEE).

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