Information-Centric Networking, A natural design for IoT applications?

Maroua Meddeb

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)
Cotutelle internationale *Ecole Nationale des Sciences de l'Informatique (ENSI)*

Présentée et soutenue le *27/09/2017* **par :**

MAROUA MEDDEB

Information-Centric Networking, A natural design for IoT applications?

JURY

HAKIMA CHAOUCHI	Prof à Telecom Sud Paris	Rapporteur
LAMIA CHAARI	MCF-HDR à l'ISIM Sfax	Rapporteur
MOHAMED MOSBAH	Prof à l'INP de Bordeaux	Examinateur
TAKOUA ABDELLATIF	MCF à l'EPT Tunis	Examinateur
THIERRY MONTEIL	Prof à l'INSA Toulouse	Codirecteur de thèse
AMINE DHRAIEF	MCF à l'ESEN Manouba	Codirecteur de thèse
KHALIL DRIRA	DR au LAAS-CNRS	Directeur de thèse
ABDELFETTAH BELGHITH	Prof à l'ENSI Manouba	Directeur de thèse

---

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
LAAS-CNRS (UPR 8001)
Directeur(s) de Thèse :
Khalil DRIRA, *Abdelfettah BELGHITH*, *Thierry MONTEIL* et *Amine DHRAIEF*
Rapporteurs :
Hakima CHAOUCHI et *Lamia CHAARI*

Abstract

The Internet of Things (IoT) is commonly perceived as the extension of the current Internet to our physical world. It interconnects an unprecedented number of sensors/actuators, referred as *things*, to the Internet. Facing the important challenges imposed by devices heterogeneity and the tremendous generated traffic, the current Internet protocol suite has reached its limits. The Information-Centric Networking (ICN) has recently received a lot of attention as a potential Internet architecture to be adopted in an IoT ecosystem.

The ICN paradigm is shaping the foreseen future Internet architecture by focusing on the data itself rather than its hosting location. It is a shift from a host-centric communication model to a content-centric one supporting among others unique and location-independent content names, in-network caching and name-based routing. By leveraging the easy data access, and reducing both the retrieval delay and the load on the data producer, the ICN can be a viable framework to support the IoT, interconnecting billions of heterogeneous constrained objects. Among several ICN architectures, the Named Data Networking (NDN) is considered as a suitable ICN architecture for IoT systems.

Nevertheless, new issues have emerged slowing down the ambitions besides using the ICN paradigm in IoT environments. In fact, we have identified three major challenges. Since IoT devices are usually resource-constrained with harsh limitations on energy, memory and processing power, the adopted in-network caching techniques should be optimized. Furthermore, IoT data are transient and frequently updated by the producer which imposes stringent requirements to maintain cached data freshness. Finally, in IoT scenario, devices are frequently mobile and IoT applications require keeping data continuity.

In this thesis, we propose a caching strategy that considers devices constraints. Then, we introduce a novel cache freshness mechanism to monitor the validity of cached contents in an IoT environment. Furthermore, to improve caching efficiency, we also propose a cache replacement policy that targets to raise the system performances and maintain data freshness. Finally, we introduce a novel name-based routing for NDN/IoT networks to support the producer mobility.

We simulate and compare our proposals to several relevant schemes under a real traffic IoT network. Our schemes exhibit good system performances in terms of hop reduction ratio, server hit reduction ratio, response latency and packet loss, yet it provides

a low cache cost and significantly improves the content validity.

Keywords: ICN, NDN, IoT, caching, cache freshness, cache replacement, routing, forwarding, mobility

Résumé

L'Internet des Objets (IdO) est généralement perçu comme l'extension de l'Internet actuel à notre monde physique. Il interreconnecte à l'Internet un très grand nombre d'objects: essentiellement des capteurs et des actionneurs. Face aux importants défis imposés par l'hétérogénéité des dispositifs et l'importance des trafics générés, la pile protocolaire actuelle va atteindre ses limites. Le réseau centré sur l'information (ICN) a récemment reçu beaucoup d'attention comme une nouvelle architecture Internet qui a un grand potentiel pour être adoptée par l'IdO.

Le paradigme ICN pourrait former la future architecture Internet qui s'est centrée sur les données elles-mêmes plutôt que sur leurs emplacements dans le réseau. Il s'agit d'un passage d'un modèle de communication centré sur l'hôte vers un système centré sur le contenu en se basant sur des noms de contenu uniques et indépendants de la localisation, la mise en cache dans le réseau et le routage basé sur les noms. Grâce à ses avantages, l'ICN peut être un framework viable pour l'IdO, interconnectant des milliards d'objets contraints hétérogènes. En effet, l'ICN permet l'accès facile aux données et réduit à la fois le délai de récupération et la charge des requêtes sur les producteurs de données. Parmi plusieurs architectures ICN, le réseau de données nommées (NDN) est considéré comme l'architecture ICN appropriée pour les systèmes IdO.

Néanmoins, de nouveaux problèmes ont apparu et s'opposent aux ambitions visées par l'utilisation du paradigme ICN dans les environnements IdO. En fait, nous avons identifié trois défis majeurs. Étant donné que les périphériques IdO ont habituellement des contraintes de ressources avec des limitations sévères en terme d'énergie, de la mémoire et de la puissance de traitement, les techniques de mise en cache en réseau doivent être optimisées. En outre, les données IdO sont transitoires et sont régulièrement mises à jour par les producteurs, ce qui impose des exigences strictes pour maintenir la cohérence des données mises en cache. Enfin, dans un scénario IdO, les objets sont souvent mobiles et nécessitent des stratégies pour maintenir leurs accessibilités.

Dans cette thèse, nous proposons une stratégie de mise en cache qui prend en compte les contraintes des périphériques. Ensuite, nous présentons un nouveau mécanisme de cohérence de cache pour surveiller la validité des contenus mis en cache dans un environnement IdO. En outre, pour améliorer l'efficacité de la mise en cache, nous proposons également une politique de remplacement du cache qui vise à améliorer les performances du système et à maintenir la validité des données. Enfin, nous introduisons un nouveau routage basé sur les noms pour les réseaux NDN/IdO afin de prendre en charge la

mobilité des producteurs.

Nous simulons et comparons nos propositions à plusieurs propositions pertinentes sous un réseau IdO de trafic réel. Nos contributions présentent de bonnes performances du système en termes de taux de réduction du chemin parcouru par les requêtes, de taux de réduction du nombre des requêtes satisfaites par les serveur, du délai de la réponse et de perte des paquets, de plus, la stratégie de mise en cache offre un faible coût de cache et finalement la validité du contenu est considérablement améliorée grâce au mécanisme de cohérence.

Mots clés : ICN, NDN, IdO, la mise en cache, la cohérence des donnée, le remplacement de cache, routage, transmission, mobilité

Contents

List of publications

International Journals

- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, Named Data Networking: A promising architecture for the Internet of things (IoT), Journal on Semantic Web and Information Systems, 2017.
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil , K. Drira, Cache freshness in Named Data Networking of Internet of things, The Computer Journal, 2017.(Minor Revision)
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil , K. Drira, AFIRM: Adaptive Forwarding based Link Recovery for Mobility Support in NDN/IoT networks, Submitted to Future Generation Computer Systems, 2017.
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, The Least Fresh First Cache Replacement Policy for NDN/IoT Networks, Submitted to Transactions on Emerging Telecommunications Technologies, 2017.

International Conferences

- M. Meddeb, M. Ben Alaya, T. Monteil, A. Dhraief, and K. Drira. M2m platform with autonomic device management service. In ANT/SEIT-Procedia Computer Science, volume 32, pages 1063-1070, Hasselt, Belgium, 2014. Elsevier.
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, Cache Coherence in Machine-to-Machine Information Centric Networks, 40th Annual IEEE Conference on Local Computer Networks, Florida, USA, 2015.
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, Producer Mobility support in Named Data Internet of Things Network. ANT/SEIT - Procedia Computer Science, Madeira, Portugal, 2017. Elsevier.
- M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, How to cache in ICN/IoT networks?. 14th ACS/IEEE International Conference on Computer Systems and Applications, Hammamat, Tunisie, 2017.

List of Figures

List of Tables

Chapter 1

Introduction

The Internet is one of the relevant innovations that have changed our lives. Its history dates back to the 1960s with the development of the first telecommunication networks with the aim to communicate two computers. Step by step, Internet services began to spread with the emergence of the e-mail then the World Wide Web. This latter is the most popular Internet application. For about twenty years, the Internet and especially the Web did not cease to evolve passing through several distinct evolutionary phases. All of them offer a network of networks providing a general infrastructure used to exchange different information around the world. We cite, in chronological order, the static Web as websites without interaction, the transactional Web as Amazon, eBay, etc. and the social web which allows people to communicate and share information about each other as Facebook, Twitter, etc. Then, the Internet has undergone the emergence of the Internet of Things (IoT) by connecting small devices to the Internet to share information with the consumer and with themselves without human intervention.

IoT is a quiet revolution that is changing the world. The connected world is becoming a reality encompassing heterogeneous constrained devices belonging to heterogeneous networks. With this evolution, everything is becoming smart able to make a decision based on the current state of the environment. The IoT is integrated into our daily lives, it now appears in various domains; Smart Home, Wearable, Retail, Smart Cities, eHealth, Agriculture, Automotive/Transportation, Industrial Automation and Energy Management.

Motivations

The IoT traffic is increasing every day due to the explosive growth of the number of "things" connected to the Internet. Recent traffic statistics made by Cisco show that the annual global IP traffic has surpassed the 1 ZB (1 zettabyte = 1000 exabytes) threshold in 2016, and will reach 2.3 ZB by 2020 [Cisco 1999]. That said, the traffic will exceed six times the world population. In other words, everyone will be connected on average to six smart objects. On the other hand, the consumers have become increasingly demanding.

They require more services to facilitate their daily lives while ensuring the security of their personal data. In addition, they desire an easy manipulation of their smart objects and especially an optimal response time. According to Nick Jones, distinguished analyst and vice president of Gartner, the world's leading research and advisory company, IoT will require a new approach. He said: "*The IoT demands an extensive range of new technologies and skills that many organizations have yet to master*," he added "*New analytic tools and algorithms are needed now, but as data volumes increase through 2021, the needs of the IoT may diverge further from traditional analytics*".

Facing these distinct particularities of IoT systems, the research community, as well as the industry, have been actively working on the adoption of new networking solutions that effectively support IoT communications. As a first step, developers have started with deploying many stand-alone IoT systems in different domains. A few time ago, they aim to develop a common, de-fragmented IoT platform. To this goal, research community has proposed several protocols and IP-based standards in order to build a unified IoT platform [Meddeb 2014]. The current Internet Protocol (IP) is a well-defined standard and enables communication with entities in different domain. These advantages make IP-based solutions reasonable and able to reach the goal. However, on a large scale, IoT poses more strict challenges as devices heterogeneity, mobility, security and scalability. Facing the inherent inefficiencies of the current Internet, such solutions become inadequate. In fact, a host-centric IP paradigm requires additional resolution systems to map application requests into IP addresses and additional protocols to support mobility. Furthermore, the unified IoT platform makes physical objects accessible to applications and often implies life safety which makes security a serious requirement. Nevertheless, IP only maintains an end-to-end security. A unified IoT platform must support a tremendous number of heterogeneous and mobile things communicating together from different organizations and domains.

The cornerstone of the IoT is *Data*. A Data to express a hot day, a massive flow of traffic on the road, to follow the market in real time, etc. This mass of data has value only when it can be translated into insights and information. This latter can then be converted into concrete actions. With such fast growth of contents and users simultaneously, the incremental changes or solutions to the current Internet architecture will hardly resist to the Internet evolution. Fearing that the current Internet architecture cannot support this evolution, many efforts have been given in recent years to develop "Clean Slate" solutions for the architecture of the future Internet. To this end, numerous researches were interested in proposing cutting-edge solutions in order to support nowadays challenges and improve the data dissemination efficiency. This issue has attracted

the attention of many researchers [Gubbi 2013]. For instance, Van Jacobson et al. have taken the initiative in [Jacobson 2009] to address this subject. They have introduced a novel paradigm, named Information-Centric Networking (ICN).

ICN proposes to shift the current complex Internet model to a simple and generic one. This paradigm considers the content as the first class network citizen. In ICN, contents are addressed and routed by their unique names and are decoupled from the address of the node storing it. In this way, consumers ask for information by its name rather than its locality address. In ICN, every content is identified by using a unique, persistent and location-independent name. This paradigm natively includes in-network caching, name-based routing, multicast, anycast, mobility and self-secured content. A wide set of IoT applications is inherently information-centric. In fact, the majority of IoT applications target data regardless of its source. For instance, environmental monitoring applications are oblivious to the information origin. ICN is a promising candidate for IoT environments. It can natively support IoT scenarios while improving data dissemination and reducing network complexity. Many researchers have recently interested on the use of ICN paradigm on the top of the *Things* layer to hide its complexity. This concept has already been investigated by several studies [Heidemann 2001, Pentikousis 2015, Baccelli 2014, Quevedo 2014a, Quevedo 2014b, Amadeo 2015, Hail 2015]. Thus, we can assume that ICN has the potential to become a key technology for data dissemination in IoT networks. Although ICN attracts manifold researches, it is still in its early stage.

Problem Statement

Despite the numerous benefits that we can obtain using ICN on the top of the IoT ecosystem, it still has several aspects to be studied. In fact, the efficiency of this paradigm strictly depends on the adopted ICN architecture, caching strategy and routing protocol while considering IoT systems requirements and specificities. Several ICN architectures have been proposed, such as DONA, NDN, NetInf, COMET, CONVERGENCE, Mobility First and PRISP [Xylomenos 2014]. All of them have proved to be a relevant solution for the Future Internet. However, we need to identify which one is the most suitable for IoT.

In order to alleviate the pressure on the network bandwidth caused by the tremendous traffic growth, ICN provides in-network caching to distribute content in a scalable and cost-efficient manner. Caching is an important component in ICN. Indeed, thanks to this feature, consumer's requests are most of the time satisfied by cache nodes rather than the producer. Consequently, the traffic load is significantly reduced. In addition,

caching affords the reduction of the required distance for data retrieval and as a consequence, the response latency diminishes. Furthermore, it increases the data availability in the network since copies of the content are stored in different locations and it avoids bottleneck caused by publishing data at a unique location. However, it is essential to ensure that the in-network caching does not increase the content redundancy in the network and that there is not useful caching nodes. The cache efficiency depends on the adopted caching strategy and its cache replacement policy. The caching strategy identifies the location of the cache nodes in the topology and the cache replacement selects the content to be evicted from the cache once this latter is full. On the other hand, the IoT is a heavily constrained environment. IoT devices are usually resource-constrained with harsh limitations on energy, memory and processing power. Therefore, the adopted in-network caching in IoT systems should minimize the access to producers while performing optimal retrieval delay and minimizing the cache cost.

In an IoT context, data are transient and frequently updated by the producer. As a consequence, copies stored in caching nodes may become out of date after a certain period of time. Some IoT applications, as eHealth, have a stringent requirement in term of data freshness. In fact, stale information retrieved from a caching node for the monitoring process may risk patients lives. Therefore, it is primordial to check the validity of the information before satisfying applications requests and furthermore, invalid content must be identified and evicted from caching nodes.

The routing protocol can also impact the performance of ICN under IoT networks. Indeed, routing and forwarding algorithms are used to redirect requests to the right content location. With the use of the in-network caching, a content may reside in many locations in the network. For that, it is necessary to be able to forward the request to the closest node that can satisfy the request. In some cases, the request cannot be responded by caching nodes due to the freshness requirement or that the content is evicted from the cache. If so, it will be necessary to send the request to the producer. However, in ICN, data objects are requested without any location information. Therefore, mobile devices are no longer reachable during and after a handoff. As a consequence, routing and forwarding algorithms should be able to support the producer mobility and redirect requests to the new locations of mobile nodes.

Contributions

The main goal of this thesis is to study the performance of the IoT networks based on the ICN concept. Our objective is to provide an efficient, performant and scalable IoT envi-

ronment by taking advantage of ICN features. Meanwhile, IoT imposes some challenges that stay in front of the ICN benefits. In this thesis, we address these challenges with the aim to improve data dissemination in IoT environments. We start by identifying an adequate ICN architecture for IoT systems [Meddeb 2017e] and then we propose four contributions.

Our first contribution addresses the in-network caching. We propose a caching strategy called *consumer-cache* [Meddeb 2015, Meddeb 2017b]. Our introduced strategy aims to increase the data availability by storing information near to the consumer. In addition, it minimizes the number of caching nodes in order to reduce the cache cost. *Consumer-cache* is designed in a way to satisfy the trade-off between the system performances in terms of hop reduction ratio, server hit reduction ratio and response latency, and the cache cost.

Our second and third contributions address the IoT data freshness requirement. We start by proposing the *event-based freshness* mechanism [Meddeb 2017b]. This proposal is responsible for checking the validity of cached items before sending them back to the consumer. This mechanism is based on time series model forecasting to estimate the validity delay of cached contents. *Event-based freshness* mechanism can successfully improve the validity percentage of retrieved data, however, it slightly decreases the system performances since it neglects near copies to find a fresh one. For this reason, in our third contribution, we propose the *Least Fresh First* (LFF) cache replacement policy which is also based on a prediction process [Meddeb 2017d, Meddeb 2017c]. LFF aims to make the cache store more efficient by keeping useful information. This policy may improve the system performance while maintaining a reasonable validity percentage. Coupling these two contributions results in good system performances with a high validity percentage.

Finally, the fourth contribution addresses the producer mobility issue. We introduce an *Adaptive Forwarding based Link Recovery for Efficient Mobility support* (AFIRM) algorithm for ICN based IoT networks [Meddeb 2017a]. This contribution is twofold, it includes a routing protocol designed to populate nodes with forwarding information and an adaptive forwarding algorithm to update the routing tables after the producer mobility. The objective of AFIRM is to be always able to reach the producer after a handover in order to reduce packet loss. Furthermore, this algorithm does not result in a high signaling overhead to support the mobility.

Manuscript outline

The rest of this thesis is organized as follows. Chapter 2 provides the background of this thesis. We first detail different ICN features namely naming, caching, routing and mobility. Then, we give an overview of the current state of the IoT and its evolution over the time. This chapter presents different ICN architectures and it discusses the adequate choice of architecture for IoT networks. Therefore, it gives the IoT requirements that should be satisfied by the adopted ICN architecture and qualitatively selects and analyzes the suitability of the Named Data Networking (NDN) architecture for IoT environment. Finally, it details an IoT scenario to explain the operation of the chosen architecture.

Chapter 3 provides the state of the art of the ICN paradigm. It presents studies that address the IoT based ICN networks idea. Then, it cites studies interested on in-network caching, data freshness, cache replacement policies and the mobility support in ICN architectures.

Chapter 4 introduces our first contribution. It first gives a focus on the NDN layer to understand its different modules and identify the ones on which this thesis will focus. This chapter addresses the in-network caching. It presents in detail caching techniques differentiating cache placement selection and cache decision policies. It presents different assumptions to satisfy fixed requirements and details our *consumer-cache* caching strategy. For a better understanding, it explains an example under an IoT scenario. This chapter finally gives the cache cost metric that will be used to evaluate the proposed strategy.

Chapter 5 provides a freshness-aware efficient in-network caching in NDN-based IoT networks. It introduces the second and the third contribution. This chapter first explains the prediction model. It gives in detail different steps and algorithms to forecast the validity delay. Then, it presents the *event-based freshness* mechanism and the *LFF* policy. For each contribution, this chapter provides the algorithm and an example under an IoT scenario to better explain the proposals.

Chapter 6 addresses the last contribution. It first identifies among different approaches that an adaptive routing is the most suitable approach to handle the producer mobility issue in NDN-based IoT networks. Then, it gives a state of the art of the forwarding and routing in NDN. This chapter introduces the AFIRM algorithm. It explains both phases concerning the forwarding tables construction and the link recovery after the mobility and as other contributions, it gives examples to better understand the two phases.

Chapter 7 evaluates our contributions. It englobes all the results of our proposed

algorithms. It first gives the simulation setup by presenting the used topology, content catalog and parameters configuration. Then, it details the evaluation metrics. To analyze the findings, we differentiate two scenarios. A static scenario to evaluate *consumer-cache*, *event-based freshness* and LFF policy and a dynamic scenario to evaluate the AFIRM algorithm.

Chapter 8 concludes this thesis and all the work carried out. It gives a recapitulation of the challenges as well as the proposed contributions and summarizes our findings. This chapter finally presents perspectives to follow up this thesis work.

Part I

General context and state of the art

Chapter 2

Information-Centric IoT Networks

Contents

2.1 Introduction

The Internet appeared, in the early 1970s, with the simple aim to connect two computers. A few years later, the usage of the Internet has evolved with the creation of new services notably, the World Wide Web (1991), BitTorrent (2002); YouTube(2005) and

Dropbox (2008). Concomitantly, many new technologies have emerged such as multimedia compression as MPEG4 (1992), and the low cost RFIDs5 and wireless technologies (1999). In addition, the number of users which access to the Web has increased, and this worldwide communication tremendously raises the network traffic. Considering this evolution, the whole concept of Internet is also evolving. Today, it is about connecting data, objects and environments. In this process of communication, the machine itself loses more and more its importance to make way to the data. This radical change is at the origin of the data-oriented communication model [Feldmann 2007, Jacobson 2009].

In this chapter, we start by introducing the Internet evolution and the emergence of the data-centric concept. Then, we define the ICN paradigm and its main features. After that, we introduce the IoT networks and we reveal their inefficiencies in the current Internet model. We provide motivations to support the ICN paradigm as a solution that may replace the present overlay and host-centric networks. As many ICN architectures have been introduced since 2007, we focus, in this chapter, on which one of these architectures is more suitable for IoT environments.

2.2 The Internet evolution

The rapid traffic growth and users expectations have raised the need for a novel communication model. This issue inspired both research community and industrial, so that a few solutions to match the new traffic pattern have been proposed such as Content Distribution Networks (CDN) and Peer-to-Peer (P2P) overlays.

Concerning P2P overlays, they allow multiple computers to communicate over a network. The particularity of their architectures lies in the fact that the data can be transferred directly between two stations connected to the network without passing through a central server. It allows all computers to play the role of client and server directly. However, decentralized peer-to-peer systems have more difficulties than client-server systems in disseminating information and coordinating the interconnection of nodes, thus ensuring low delays in requests. CDNs was the most solicited solution [Pathan 2007]. They provide content distribution functionalities built at the application layer on top of the current Internet infrastructure. Some deductions extracted from the last Cisco Visual Networking Index (VNI), which analyzes the traffic composition, affirm that the Internet is shifting to an effective mobile Internet. Indeed, mobile data will represent the 61% of the total traffic by 2018. According to Cisco, CDNs are identified as the most suitable approaches for mobile data dissemination. They expect that 67% of all mobile traffic will cross CDNs by 2018 [Cisco 1999]. Despite their importance, the CDNs have

shown some limitations. The main drawbacks are related to the generated cost. In fact, CDNs are very expensive solutions especially if deployed at large scale. Moreover, they give rise to a high cost in terms of resources (bandwidth, storage, nodes distribution). The deployment of servers and the choice of their locations are not well controlled due to a lack of collaboration between the different Internet access operators. In addition, current CDNs are mono-specialized. They distribute content according to agreements with the original data providers. Finally, there is no collaboration between the different CDN suppliers and each operates individually.

Despite their advantages, CDN and P2P do not give a radical solution to deal with the fundamental issues caused by the current Internet architecture. Indeed, these approaches are overlaid on top of the current networking architecture. With such fast growth of content and users simultaneously, the incremental changes or solutions to the current Internet architecture will hardly resist to the Internet evolution. Therefore, many efforts have been given in recent years to develop "Clean Slate" solutions for the architecture of the Future Internet [Feldmann 2007, Jacobson 2009, Ahlgren 2012, Xylomenos 2014]. The ICN paradigm is the cornerstone of all proposed solutions.

2.3 Information-Centric Networking

ICN is a recent communication paradigm. Its main goal is to improve the data dissemination efficiency in the network to better adapt it to the current Internet's usage. The ICN consists of redesigning the Future Internet architecture, it constitutes a shift from a host-centric view of the network to a content-centric one. In this concept, the information or the content is considered as the primary entity rather than the host as in current networks. Therefore, the focus is on WHAT to communicate rather that WHO to communicate with. This section introduces the information-centric approach from a generic perspective by describing the main common components to a specific view by presenting the proposed architectures and their respective design choices. The staple key concepts of ICN are the information naming, the information delivery, the in-network caching and the mobility.

2.3.1 Information Naming

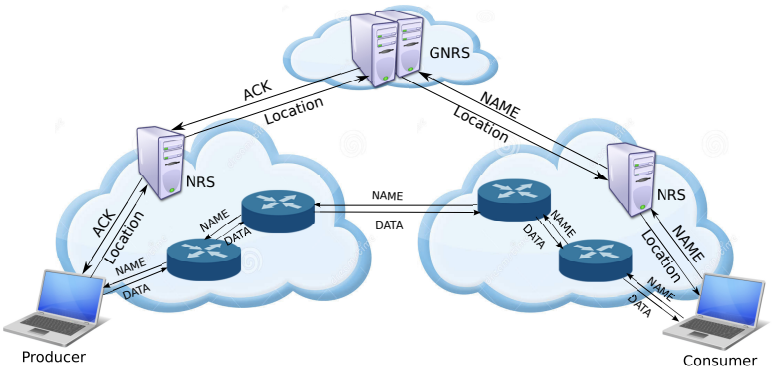
Naming data objects is an as important pillar for ICN as addressing hosts is for today's Internet. According to this new paradigm, the consumer requests a content by its name instead of using its network localization. That being said, every content must be identified by using a unique, persistent and location-independent name. Depending

on the architecture, names can be flat or hierarchical and may or may not be human-readable [Detti 2013, Xylomenos 2014]. The hierarchical namespace has a structure similar to current URLs and much like DNS names. Names are a sequence of strings separated by "/". For example, *Foo.com/video1.mp3*. The advantage of hierarchical names is that they can contain other information as the version and the chunk number. In fact, a content can be divided into many chunks. $Foo.com/video1.mp3/s_1$ identifies the first segment of the content. On the other hand, contents can be updated by the producer so we can differentiate different versions using the name like $Foo.com/video1.mp3/v_1$. A requested content named *Foo.com/video1.mp3* can be matched by an information object named $/Foo.com/video1.mp3/v_1/s_1$, which means the first segment of the first version of the requested data. After receiving this information object, the consumer can directly ask for the next data segment or for a newer version. A flat name is represented by a string as *0x3fb889fffa*. As we can see, names can be human-readable, which facilitates consumers requesting desired content. On the other hand, consumers can not understand a non human-readable name. However, these names are self-certifying. In fact, names can be encrypted with the data itself and its producer which make them non human-readable. This group of namespace has the security advantage. Since requests can be satisfied by nodes other than trusted origin producers, consumers should be able to verify the integrity and the provenance of received data by means of "security" information included in the data object. To recapitulate, there are four namespace classes and the choice depends on the applications and consumers expectations. The decision whether to use flat versus hierarchical names either human-readable or self-certifying mainly impacts the scalability of the ICN routing plane.

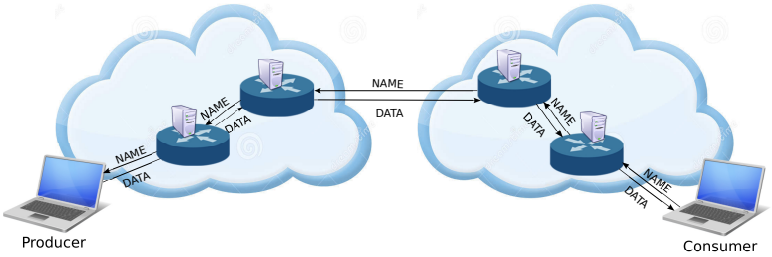
2.3.2 Routing

The routing is a question of how requests are routed towards the producer through the network and then how data are routed back to the consumer. There are two general approaches in ICNs to handle routing requests, both strongly depending on the properties of the object namespace, in particular, the aggregability of object names [Ahlgren 2012]. ICN implements the name resolution routing and name-based routing depicted in Figure. 2.1.

With the name resolution method, the data retrieval is performed in two steps as shown in Figure. 2.1a. First, the content name is translated in one or many locators if exists. These latter are the current topological locators of the requested content in the network, it can be the producer address or a cache node that maintain a copy of the desired content. The entity that stores these locations information is called *Name*



(a) Name Resolution Routing



(b) Name-based Routing

Figure 2.1: Routing approaches

Resolution System (NRS). The NRS stores a binding between content names and their current locators. NRSs are arranged in a hierarchical manner and each one covers a specific area of the network, as a consequence, the consumer must redirect the request to its dedicated NRS to retrieve the location information. If the NRS does not have the location information, it redirects the request to the Global NRS of the upper level. Once having this information, the consumer can then directly forward the requests towards the optimal destination based on routing protocols. Different NRSs in the topology are populated thanks to the signaling messages coming from producers to announce a content availability. It is worth to note that each producer notifies only its dedicated NRS.

Unlike named resolution routing, the name-based routing method, presented in Figure. 2.1b, relies on just one step to retrieve a content. This approach is based on the name hierarchy to route requests and directly forward them to producers without requiring to resolve names to locators in a previous step. At each node towards the producer, the request is sent to the next interface that matches the longest prefix. This means that each node should be aware of a part of the routing information. After the producer

has received the request, the data is routed back to the consumer through the request reverse path.

2.3.3 Caching and storing

As information can be named independently from the location of the respective producers, data can be stored everywhere in the network. Therefore, copies of the same content stored in different locations in the network, are considered as a unique content. It is about the in-network caching, which is a major building block of ICN. It represents the most common and important feature of ICN architectures. It was introduced to alleviate the pressure on the network bandwidth and consequently improve the transmission efficiency in content dissemination [Zhang 2013]. As shown in Figure. 2.2, consumer 2 sends a request to retrieve the content named X. While sending the response, node 2 and node 6 store a copy of the content. Then, when consumer 1 or 3 requests the same content X, the request is satisfied by nodes 2 or 6 respectively.

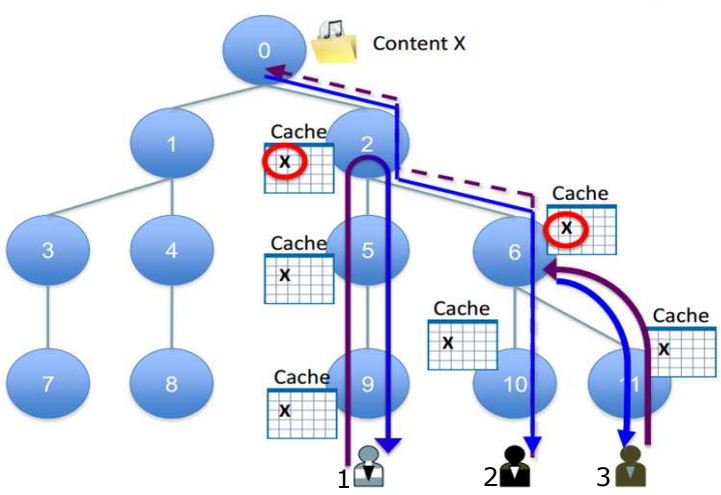


Figure 2.2: In-network caching in ICN

The caching concept, which is not a revolutionary term, has been widely used on the Web, P2P systems and CDNs. However, in ICN, the in-network caching is more prominent and more challenging than in the already existing caching systems. First, it is transparent and does not require any specific application to cache content. Second, it is ubiquitous since any ICN nodes can be a cache [Zhang 2013].

On the other hand, it is worth to note that the cache size is limited and once the cache is full, a cache eviction is performed to allow caching new items. Consequently, the

in-network caching imposes the establishment of a *cache replacement policy*. According to the replacement policy, one of the cached contents is selected and deleted.

2.3.4 Mobility

Another immediate consequence of naming data objects is that even if contents change their locations, they will be always reachable since their names are location-independent. The mobility in ICN can be classified into *Consumer* mobility and *Producer* mobility [Zhou 2014]. The former can be easily handled. In fact, when a consumer moves, it just needs to re-express its interest for a data object from the new location. Either with name-based routing or NRS-routing, producer mobility may cause the loss of request packets when the request is not satisfied by a cache node. In the case of name-based routing, the request will be redirected hop by hop according to the routing information in each node. When a mobility occurs, the routing information needs to be updated in all nodes belonging to the request path. In the other case, the routing is based on the location retrieved from the corresponding NRS. After a handover, NRS systems should be updated to match names to new locations.

2.4 The vision of Internet of Things (IoT)

Despite the diversity of research on IoT, its definition remains fuzzy. Authors in [R. et al., 2015] addressed this challenge by collecting most of the definitions and architectural models proposed for IoT in the literature and they gave a definition that addresses all the IoT's features. IoT may refer to a transition state, an everyday object, a virtual identity, an extension of the Internet, an enabling framework, or simply an interconnected world as the definition changes with relation to the speaker point of view. In fact, Cisco admits that IoT was born sometime between 2008 and 2009 when more "things" were connected to the Internet than people. According to IoT Special Interest Group (SIG), IoT is a transition state where things will have more and more information associated with them and may have the ability to sense, communicate, network and produce new information, becoming an integral part of the Internet. In the viewpoint of US National Intelligence Council (NIC), IoT refers to everyday objects that are readable, recognizable, locatable, addressable, and/or controllable via the Internet. It can focus on the virtual identity of the smart objects and their capabilities to interact intelligently with other objects, humans and the environment. According to the IETF, the IoT is an extension of Internet technologies to constrained devices, moving away from proprietary architectures and protocols. Most of these definitions agree with the fact that IoT is a new phenomenon

that is changing the world in which we live, trying to connect heterogeneous devices with limited resources in terms of battery, memory and computing power. Managing and analyzing data from such hyper-connected networks would require robust architecture and protocols. In this section, we first analyze the applicability of the Internet protocol in an IoT environment. Then, we present the IoT standardization effort.

2.4.1 IoT under the TCP/IP stack

IoT aims to extend the Internet to physical objects in order to allow connectivity and interoperability between them. To this end, these IoT objects should embed an IP network protocol stack. As we have already mentioned, IoT devices are heavily constrained and as such embedding a regular TCP/IP stack, in such an environment, is a serious challenging issue. First, IoT applications are able to generate a high throughput compared with the low data rate and low power of IoT devices coupled with lossy wireless links. Second, the tremendous number of heterogeneous IoT devices makes the management task very complex since all devices need to be automatically managed. Moreover, the mobility requirement supported by IoT devices makes this task more and more arduous. Finally, TCP acknowledges every received segment, which pushes the network to undergo an additional signaling cost. For these reasons, the use of regular TCP/IP on IoT systems is impractical and problematic [Duquennoy 2011, Montavont 2014, Zhang 2016, Msadaa 2016].

2.4.2 IoT standardization effort

Until now and despite the numerous proposals, there is still no consistent common architecture for the IoT. However, some protocols are believed to be part of a future unified IoT architecture [Ishaq 2013]. In Figure. 2.3, we present a protocol stack that covers prominent protocols. In the following, we detail these protocols, starting with the application layer down to the physical layer.

Concerning the application layer, the Internet Engineering Task Force (IETF) working group, named COnstrained RESTful Environments (CORE), proposed the application protocol Constrained Application Protocol (CoAP) to be used by constrained devices (low power, low data rate) on constrained networks. CoAP can be embedded on very constrained nodes since messages are characterized by a short 4-byte header. For the transport layer, CoAP is by default used by the UDP transport protocol. Datagram Transport Layer Security (DTLS) is integrated to support security. Moving to the network level, the IETF Routing Over Low power and Lossy networks (ROLL) working group has proposed a routing protocol called Routing Protocol for Low-Power and Lossy

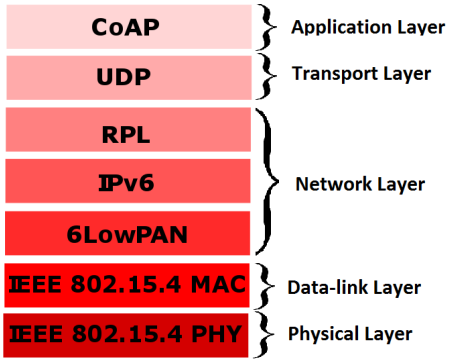


Figure 2.3: IoT standardization effort

Networks (RPL). RPL is adapted to IoT environment since it natively supports multi-party communication pattern. On the network layer, the IPv6 is considered as a viable addressing solution in the IoT. The IoT addressing scheme must support the huge number of IoT objects. Furthermore, with the heavily constrained environment, addressing protocol should be the less complex possible. In addition to the addressing protocol, the 6LoWPAN protocol is used to encapsulate and compress the IPv6 header in order to adapt its size to the IoT environment (from 40 bytes to 2 or 3 bytes). Finally, concerning Link Layer and Physical Layer, the IEEE has proposed the IEEE 802.15.4 MAC and the IEEE 802.15.4 Phy protocols with an ultra-low-power consumption [Msadaa 2016].

Although this stack was specifically designed for IoT systems, it has several weaknesses. The first gap is caused by IP addresses which have a dual semantic: they are simultaneously used as a locator and as an identifier. However, in an IoT environment, an identifier does not necessarily refer to a particular device: it can refer to an equipment, a content or a service. Consequently, this stack still requires a name resolution system which induces important additional IoT flows. Another significant problem of this stack is the relatively large signalization generated: Baccelli et al. highlighted in [Baccelli 2014] the heavy burden of the signalization generated by this stack for constrained equipment with a memory not exceeding 10 kilobytes. Finally, IoT devices often provide sensors values that cover large areas. This stack does not allow access to the optimal content: Get the nearest value or take advantage of the already requested values. This is because the paths created by the stack do not take into account any caching mechanism provided by IoT equipment.

Current IoT systems rely in general on a centralized communication model in which the IoT servers are responsible for handling and relaying all distributed signaling within the system. It is well known that centralized systems can surely yield strong optimization

but, at the same time, critical entities of the system that act as concentrators may suddenly become single points of failure, thus lowering the fault tolerance of the entire system. ICN has shown that it can enhance IoT networking and data dissemination. Moreover, it can fill the gaps in the stack CoAP/RPL/6LowPan/802.15.4 and reduce its complexity. In fact, the IoT applications are essentially centered on information since they process data independently of equipment that generates them. In ICN, thanks to the naming scheme and in-network caching, requested contents are addressed by their unique name and can be satisfied by any cache holding it. As a consequence, caching content will impact the lifetime of the IoT devices' batteries: a request may be satisfied by an active node while the information producer remains in its sleep mode. Finally, security is paramount in an IoT context, as this is an extension of Internet technologies to the physical world. ICN addresses this requirement and targets to secure the contents themselves rather than securing the channels connecting the equipment with each other. ICN is a promising candidate for IoT environment. It can natively support IoT scenarios while improving data dissemination and reducing network complexity.

2.5 Which ICN architecture for IoT?

In this section, we first detail ICN approaches and then we qualitatively compare them and investigate which solution can fully satisfy the IoT requirements.

2.5.1 ICN architectures

In the following subsections, we introduce and illustrate ICN approaches with the purpose of general understanding them. We present ICN initiatives according to their publication chronological order as shown in Figure. 2.4.

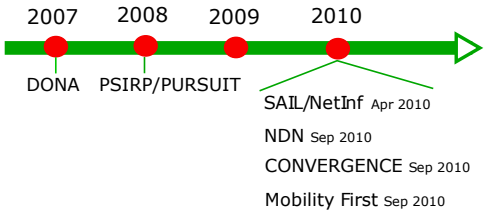


Figure 2.4: Timeline of key ICN milestones

2.5.1.1 Data-Oriented Network Architecture (DONA)

Data-Oriented Network Architecture (DONA) [Koponen 2007] is one of the first complete architectures. It radically changes naming from hierarchical URLs to flat names. In DONA, contents are published in the network by the sources. Nodes that are authorized to get data, register to the Resolution Handlers (RH). In this architecture, request packets, named *Find* packets, are routed by name toward the appropriate RH. The *Data* packet is sent back through the reverse path enabling caching. The data can also be sent over a direct route. DONA supports the on-path caching. Cache nodes are RHs that can decide whether to cache or not the content. For the mobility support, mobile subscribers must issue new *Find* message from their current locations. Concerning publishers mobility, they can re-register their information when changing their locations. Names in DONA are self-certifying; consumers receive in the *Data* packet the public key of the owner and a signature of the data itself. They can verify that the received data matches the requested name.

2.5.1.2 Publish-Subscribe Internet Routing Paradigm (PSIRP)

The Publish-Subscribe Internet Routing Paradigm (PSIRP) [Ain 2009] project and its continuation, the Publish Subscribe Internet Technology (Pursuit) [FP7 2010c] project, introduced a publish-subscribe protocol. This architecture is composed of three major components; the *rendezvous* system, *topology manager* and *forwarding* mechanism. An object in PURSUIT is identified by the pair (the *scope* ID and the *rendezvous* ID). They both belong to a flat namespace. The publisher sends a *Publish* message to its *Rendezvous Node*(RN) to announce an information. This *Publish* message will be routed to the RN assigned with the corresponding scope ID. In the other part, the subscriber issues a *Subscribe* message to its local RN that will be also routed to the RN assigned with the corresponding scope ID. Publications and subscriptions are matched in the *rendezvous system*. The matching procedure results in a Forwarding Identifier (FI) sent to the source. The latter forwards the data to the subscriber. PURSUIT can support the in-network caching. However, it may not be very effective. In fact, cache nodes act as publishers; they must publish available information. Mobility is facilitated in PURSUIT since both subscribers and publishers always send messages for each operation, so although they change their locations, the new location will be communicated for future operations. PURSUIT greatly supports security. The use of flat names allows self-certifying names. In addition, this approach uses the Packet Level Authentication (PLA) technique to assure data integrity and confidentiality by encrypting and signing

individual packets.

2.5.1.3 COntent Mediator architecture for content-aware nETworking (COMET)

The COntent Mediator architecture for content-aware nETworking (COMET) [FP7 2010a] considers a Content Mediation Plane (CMP) that is the core component of its architecture. The CMP mediates between the network providers and information servers. It has a global idea about information and infrastructure. For this approach, there is no precise defined naming scheme. However, unlike other ICN approaches, COMET allows the location preferences to be explicitly included in data. In COMET, publishers that want to publish contents can send a *Register* message to the corresponding Content Resolution System (CRS). The CRS gives a name to the content, stores its actual location (IP address) and sends *Publish* messages to propagate the content to its parent CRS. Subscribers issue a *Consume* message to retrieve a content. This request is sent to its local CRS and it will be propagated upwards in the CRS hierarchy until it reaches a CRS that maintain information about the requested content. When a match is found, the request follows the location stored in the CRS to reach the publisher. COMET supports the in-network caching. This approach holds user mobility by using specialized mobility-aware Content-aware Routers (CaRs), placed at the edge of the networks, to track the mobility of users and information.

2.5.1.4 Scalable and Adaptive Internet Solutions (SAIL)

The Scalable and Adaptive Internet Solutions (SAIL) [FP7 2010d] has a very wide scope. We will focus here on the Network of Information (NetInf) elaborated within this work and introduced in [Ahlgren 2010]. The NetInf employs a flat name. This architecture provides us with two models for retrieving data; name resolution or name-based routing. Under the first model, sources publish data by registering it to the Name Resolution Systems (NRS) by a name/locator binding. The receiver can send a *Resolve* packet with the data name to the NRS to get the best available source. Then, it can get a copy of data from this source. With name-based routing, the receiver directly sends a *Get* message with the data name that will be routed by name toward a copy of data. When a copy is found, the data will be returned to the receiver via the request reverse path. This approach supports request and objects caching. When a node receives a Get request, it can decide to store the request in a pending interest table. NetInf supports both subscribers and publishers mobility. When a host changes location, it updates the topological information in the NRS where it is registered. Then, a notification is sent

to all hosts that communicate with the mobile one. As in DONA, the NetInf can check the name-data integrity using self-certifying names.

2.5.1.5 Content Centric Networking (CCN)

The Content-Centric Networking (CCN) [Jacobson 2009] architecture is one of the pioneering ICN structures. The Named Data Networking (NDN) [NSF 2010b], funded by the US Future Internet Architecture program, implements the CCN architecture. In NDN, names are hierarchical and may be human-readable. NDN is based on three system elements; Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB) table. Consumers issue *Interest* messages to request information objects. The request is routed by name. CS acts as a cache in which the received data are cached. When a request hits a cache, a CS lookup is performed. In the case of a cache miss, the *Interest* packet is sent to the next hop according to the FIB, and the interface of the incoming *Interest* is appended to a set of interfaces interested by that chunk in the PIT. The FIB contains, for each content, an entry that points to the right output interface, the request is redirected to the interface with the Longest Prefix Match (LPM). In the case of a cache hit, the corresponding data is sent back according to interfaces stored in the PIT. When the request is satisfied, the corresponding entry in the PIT is then removed. NDN approach natively supports in-network caching using the CS. Like other architectures, NDN supports consumers mobility by sending new *Interest* messages from the current location. However, when a producer moves, all FIBs must be updated. Concerning the security, *Data* messages in NDN contain a signature and the signer public key as a metadata. So, when a consumer receives a data, it can check its integrity.

2.5.1.6 CONVERGENCE

The CONVERGENCE, introduced in [FP7 2010b], has many similarities with NDN. In CONVERGENCE, objects are presented with a *Versatile Digital Item* (VDI). VDI names can be flat or hierarchical. In this architecture, subscribers send *Interest* messages to retrieve a VDI. The request is routed by name toward publishers. However, nodes do not maintain name-based routing information for each name prefix. If there is no routing information for a corresponding name, the node consults an NRS to find out how to forward the *Interest*. Publishers respond with *Data* messages that follow the reverse path. Obviously, CONVERGENCE allows caching. Subscriber mobility is supported by sending new *Interest* messages from the new location. This approach proposes to check

the security at subscriber level when it receives a *Data* message. The security is checked using the digital signature included in the data.

2.5.1.7 Mobility First

The Mobility First [NSF 2010a], introduced by the US Future Internet Architecture program, proposes an ICN architecture that targets the devices mobility problem. The aim of this architecture is to address host, information and entire network mobility. The Global Name Resolution Service (GNRS) handles the host mobility by updating its entries when an object changes its point of attachment. Objects in Mobility First are identified with a Global Unique Identifier (GUID). GUIDs are flat. In this architecture, a publisher, that wants to make information available, asks the GNRS for a GUID and registers it with the given name. The subscriber issues a *Get* message that includes the GUID to its Content Router (CR). The CR asks the GNRS for the network address of the data. The GNRS replies with a set of addresses. The CR selects one address to which it sends a Get message. The request is routed through the CRs. Each CR can consult the GNRS to update the address of the data. The publisher follows the reverse path to send its response. Mobility First supports caching via CRs. Security, in this architecture, is maintained by self-certifying names. Furthermore, the user can frequently request new GUID to avoid profiling.

2.5.2 IoT fundamental requirements

We start by presenting the stringent features that an IoT architecture requires from ICN approaches. IoT contents describe the behavior of a smart object, or report the state of an environment or a person. These contents are identified with URIs (Uniform Resource Identifier) under a hierarchical form. In most cases, names are human-readable to make them more easily identified by consumers. According to applications, names can be logically or geographically aggregated.

IoT supports a receiver-driven protocol to disseminate the information. This routing approach transmits data from a transmitter to many receivers. When a requested content is routed to its requester, it can cross a router in which there is a pending request for the same data. In this case, a copy of this data is redirected to the second requester, and the content continues its way towards its destination and so on.

The IoT communication model is based on a pull paradigm. In the first phase, a node requests a content. Then, in the second phase, the node should receive this content. In some other cases, the IoT traffic pattern can include an observe/push paradigm whereby

a consumer can make a subscription to a specific content. In this way, this consumer will receive a copy of the content after each update. In these two paradigms, it is the consumer who solicits the producer for a content. There are systems other than IoT where producers always publish their contents on the Internet without being asked for any content.

Iot systems consider a multi-party communication pattern. This model can represent multi-consumer or multi-source communications. In multi-consumer communications, the server, that maintains an entire smart environment, can receive many different requests from different consumers interested in this environment and the state of its appliances. Concerning the multi-source communications, it supports the transmission of multiple separate requests from a consumer to retrieve data from different producers.

Currently, the used IoT link has a very small maximum frame size, such as IEEE 802.15.4, Bluetooth, etc. In addition, IoT devices have memory constraints. Consequently, IoT scenarios should support data fragmentation. This latter consists in splitting the data into segments in such a way that each segment satisfies the system constraints.

The majority of IoT devices, such as smartphones, laptops, and embedded objects can be mobile. Devices, such consumers or producers, may easily switch between networks. To deal with this feature, the entire network should be notified when a device changes its location. This modification must be considered in next configurations or tasks.

IoT was designed to operate in a completely trustworthy environment. User authentication and data integrity and privacy are necessary for this kind of networks. Some application domains in IoT, such as smart home or eHealth, concern users' personal private lives.

Concerning the routing approach, we should focus on the two routing scenarios proposed in ICN to select the most efficient for IoT networks. The name resolution method presents some shortcomings that impact nodes memory. In fact, in addition to the classical IP routing table existing in each node, name matching tables have to be stored in NRS nodes. Moreover, since these latter are organized following a hierarchy, the NRSs volumes become more important with higher levels until storing the information of all the content in the top layer. This approach also requires a heavy signaling process to retrieve contents which increases the communication overhead. On the other hand, assuming that routing updates should be flooded throughout the whole network, the name-based routing raises the communication overhead. Nevertheless, by comparing it to the name resolution method, this one is memory saving since it just requires to store in nodes the mapping of content names to the network location. Furthermore, in

opposition to the name resolution routing, this approach improves the resilience against request failure since the resolution process is distributed. In order to reduce the energy consumed by radio transmissions compared to basic routing approaches, it is more efficient to consider the name-based routing using the hierarchical form of names.

2.5.3 ICN in IoT environment: suitability analysis

We discuss here the IoT requirements satisfied by each of the ICN architectures and we summarize the comparative synthesis in table 2.1. Only NDN and CONVERGENCE architectures use hierarchical human-readable names and allow data fragmentation. One of the advantages of hierarchical names consists in easily allowing multi-source communication patterns. Since producers share a common name prefix, consumers can ask different producers by sending a request with using just its prefix name. For instance, a Get request with /homeID/ name is sent to get information about the entire house. As a result, only NDN and CONVERGENCE can permit multi-part communication patterns. However, in addition to the security treated by all initiatives expect COMET, the mobility is supported by all ICN architectures. DONA, SAIL, PURSUIT, NDN and Mobility First can support multicast channels by allowing Get messages to be cached in intermediate hosts. When a Get message requesting the same name reaches a host, the latter directly returns the data to the requester. Consequently, these architectures are receiver-driven approaches. Routing in COMET, PURSUIT, CONVERGENCE and Mobility First is based on a name resolution paradigm. All these architectures use the Name Resolution Systems to forward packets in the network. DONA, SAIL and NDN are the only architectures that define a name-based routing. In DONA, SAIL, NDN and CONVERGENCE architectures, subscribers interested in some content, do so by sending a Get message. The producer, or eventually a cache, replies to this request. As such, these architectures follow a pull-based communication model. In summary, only the NDN architecture satisfies all of the IoT requirements. We may then conclude that the NDN approach is the most appropriate ICN architecture for IoT systems.

2.5.4 NDN for IoT

The features of the NDN architecture make it a promising solution to fit the peculiarities and requirements of the IoT environment. For the sake of clarity, Figure.2.5 sketches the NDN architecture under IoT deployment. $Consumer1$ is interested in content $/Home/room1/Tmp$, so it sends an *Interest* packet towards $n1$. This latter, upon receiving the Interest, checks its CS to verify if there is a copy of the asked content in

Table 2.1: Comparative table

	DONA [Koponen 2007]	PURSUIT [FP7 2010c]	COMET [FP7 2010a]	NetInf [FP7 2010d]	NDN [NSF 2010b]	CONVERGENCE [FP7 2010b]	Mobility First [NSF 2010a]
Hierarchical/ human-readable name			(not defined)		✓	✓	
Receiver-driven	✓	✓		✓	✓		✓
Pull-based paradigm	✓			✓	✓	✓	
Multi-party communication pattern					✓	✓	
Name-based routing	✓			✓	✓		
Fragmentation					✓	✓	
Mobility	✓	✓	✓	✓	✓	✓	✓
Security	✓	✓		✓	✓	✓	✓

the cache. In the case of a cache hit, the data is sent back to $Consumer1$. In the case of a cache miss, a PIT check is performed. When the content name is found as an entry in the PIT, this means that there is a node other than $n1$ that already ask for the same information. Then, the *Interest* packet is discarded and another entry is added in the PIT with $n1$ as the node that sends this request. If there is not a match in the PIT, the *Interest* is redirected to the next node as prescribed by its FIB. If the FIB does not give any information about the next hop, that said there is a routing problem and the *Interest* is deleted.

IoT promises to connect billions of objects to the Internet. To this end, a few proposals built a unified host centric IoT platform. However, such solutions are inadequate to address the heterogeneity and mobility challenges of the IoT environment. The NDN philosophy is rather generic and can match any device or application specificities. This is due to the variable namespace, so developers are free to conceive a naming scheme that fits the constraints of their environment. By leveraging named contents and name-based routing, NDN offers easy, robust and scalable data retrieval. This is due to the use of hierarchical names and PIT and FIB structures that form a smart forwarding fabric. Consequently, we will no longer need the IP address assignment procedures for content search and retrieval which are a communication burden, especially in large networks.

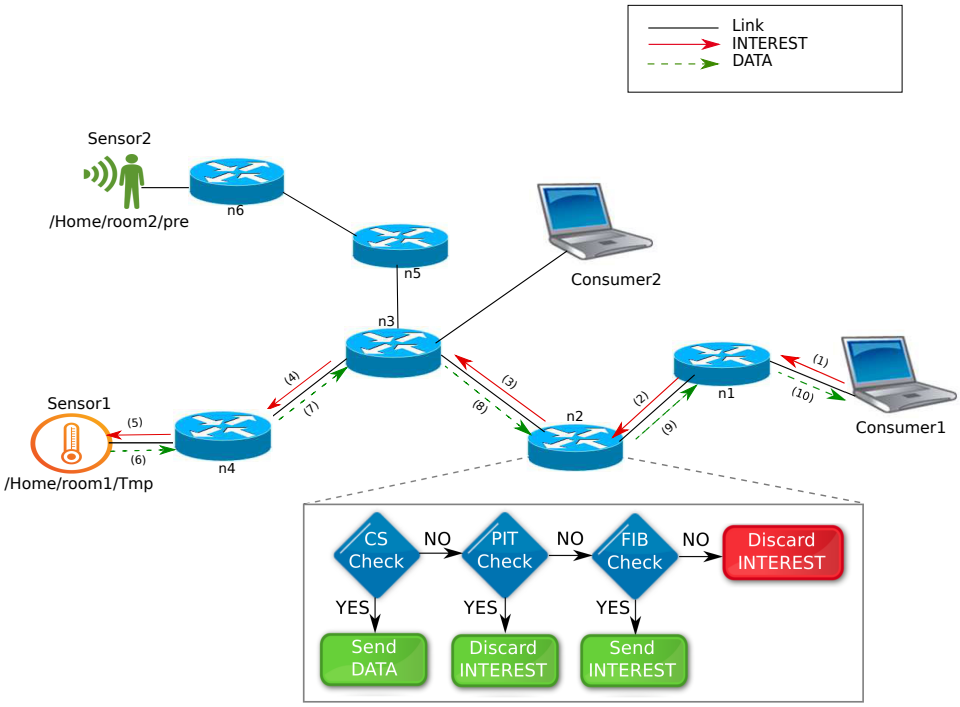


Figure 2.5: Forwarding operations in NDN

To deal with the problem of massive data access, NDN provides Interests aggregation within the PIT structure and in-network caching. In fact, NDN offers the *multicast* feature as the PITs can identify multiple requests for the same content and forward just a single Interest. In addition, consumers can retrieve cached contents from intermediate routers rather than from their sources. Moreover, with the native *anycast* feature, NDN can avoid constrained nodes (as unavailable nodes or low battery routers, etc). Requests are rather routed to more capable routers. In-network caching can also take into consideration constrained devices by preventing caching to be performed at these constrained devices. NDN also supports many providers to one consumer communications thanks to the natural name aggregation. Coupled with the *anycast*, this communication model improves significantly the energy efficiency of the overall network. To address the mobility requirement in IoT environments, including things and users mobility, IP networks introduced the mobile IP patches. Such a solution is not recommended because of its heavy forwarding overhead and long induced delays. Thanks to the use of location-independent names and receiver-driven communications, NDN natively supports consumer mobility. Nevertheless, producer mobility requires the updating of the FIBs. However, in-network caching can alleviate this updating process. We conclude that NDN functionalities as

Named Data, In-network caching, Name-based routing, Multicast and Anycast can natively support IoT requirements. We summarize, in Table 2.2, which NDN functionality can intrinsically satisfy which IoT requirement.

Table 2.2: Main IoT requirements and native NDN support

	Named Data	In-network caching	Name-based routing	Multicast	Anycast
Scalability	✓	✓		✓	✓
Quality of service		✓	✓	✓	✓
Energy Efficiency		✓		✓	
Heterogeneity	✓				
Mobility	✓		✓		✓
security	✓				

2.6 Conclusion

Chapter 2 was dedicated to defining the general concepts of this thesis, notably, the ICN paradigm and IoT networks. We also showed, in this chapter, that ICN can be an efficient network for IoT scenarios that can improve the data dissemination and deal with current network limitations. We finally demonstrated that NDN is the most suitable ICN architecture for IoT systems. In the next chapter, we will present the related research close to our work. In particular, we will further detail studies that focus on in-network caching, cache freshness, cache replacement policies and mobility in NDN IoT networks.

Chapter 3

State of the art

Contents

3.1 Introduction

This thesis, as mentioned in chapter 2, studies the use of ICN in the context of IoT environment. This issue has attracted some researchers in the last few years. In this chapter, we present the state of the art. We detail researches that highlight the general benefits that ICN could provide to IoT networks. In a second party, we are particularly interested in the in-network caching, the cache freshness, the cache replacement policies and the mobility support.

3.2 Information-Centric Networking for Internet of Things applications

The deployment of IoT systems enabling ICN features has started to gain momentum within the research community. An interesting contribution in this field was proposed

by Pentikousis et al. in [Pentikousis 2015]. The authors presented several ICN baseline scenarios including IoT applications.

Some studies have focused on IoT requirements that ICN architectures can natively support without setting which architecture is the more appropriate for IoT scenarios. An interesting contribution in this field was recently proposed in the Internet Draft by the ICN Research Group in [Zhang 2016]. The authors discussed the main IoT requirements and ICN challenges to realize a unified framework. Authors in [Xu 2014] argue that ICN is the ideal candidate architecture for IoT systems. In [Amadeo 2016b], Authors detail IoT requirements and present several possible motivations for the introduction of ICN in the context of IoT.

The beneficial effect of the named data on IoT is identified in [Heidemann 2001] by Heidemann et al. Authors discussed that content naming is considered as the building block of the ICN by making the information easily and uniquely identified in the network. Moreover, they proved that naming, as one of the ICN features, makes the deployment of IoT in ICN more effective.

Lindgren et al. in [Lindgren 2016] describe which Internet architecture would de be beneficial for a specific IoT context. They explain that for some IoT application a host-based network is more efficient that an information-based one and they indicate potential modifications of ICN in order to improve efficiency and scalability for IoT applications.

In conventional ICN, contents are requested by their names. That said, consumers should know the exact identifiers beforehand. In [Kurita 2017], authors address this issue and propose an ICN extension named Keyword-Based Content Retrieval (KBCR). Requests are spread to multiple data related to the keywords. Then, all responses are merged into one response.

As concluded in chapter 2, NDN is the most convenient ICN architecture for IoT systems. Indeed, several studies have argued that NDN is the most convenient approach for IoT systems such as [Francois 2013, Amadeo 2014a, Baccelli 2014, Hail 2015, Amadeo 2015, Amadeo 2016a, Amadeo 2014b]. So far, research that targets to use NDN for IoT is still in its infancy.

Authors in [Francois 2013] aim to optimize the traffic within a CCN for IoT network. In this article, François et al. promote the usage of CCN for IoT due to the advantage of the hierarchical routing by leveraging a scalable hierarchical naming scheme and the security mechanisms and caching provided by design. The simulations show that the proposed optimization algorithm outperforms existing ones.

In [Amadeo 2014a], authors address the design of a high-level NDN architecture that meets IoT challenges. In this paper, they present the main benefits of NDN and discuss

how this architecture can support IoT main requirements.

The work in [Baccelli 2014] constitutes the first study of ICN in a real IoT deployment. Bacceli et al. defended the NDN architecture that is very effective in IoT scenarios. They also proved that this architecture satisfies IoT requirements.

Amadeo et al. in [Amadeo 2015], defined NDOMUS (Named Data netwOrking for sMart home aUtomation) as a framework based on the NDN-IoT architecture tailored to the smart home domain. This paper presents many use-cases. It also evaluates the proposed framework by calculating the number of the transmitted packets.

The work presented in [Amadeo 2016a], proposes an ICN-based solution for resource-constrained devices in M2M domains. For the design of the M2M-ICN Layer, authors refer to NDN among several information-centric architectures. The proposal aims to use the OM2M [Alaya 2014] open source implementation of the ETSI M2M standard to deploy NDN Gateways.

There is also a study that considers the ICN approach in a specific IoT application domain which is Wireless sensors networks [Amadeo 2014b]. Authors, in this paper, present the main features of wireless sensors, mobile and vehicular ad hoc networks and identify the benefits of CCN is such environment.

Other few studies considered the ICN approach in specific IoT application domains as Vehicle-to-Vehicle (V2V) communications [Wang 2012b], Wireless sensors networks [Dinh 2013, Ren 2013, Teubler 2013, Amadeo 2013], Smart Home [Amadeo 2015] and Smart Grid [Katsaros 2014].

3.3 In-network caching

In-network caching is one of the inherent features in ICN paradigm treated in this thesis. Caching is not a revolutionary concept; it has already been widely used in the Web [Breslau 1999], P2P overlays [Saleh 2006] and CDN [Vleeschauwer 2011]. The common logic is to give to specific network entities the ability to store some Internet contents and consequently satisfy corresponding requests. Though similar at their first glance, there are many inconsistencies between the different approaches [Zhang 2013]. First, in ICN, caches are application-independent which offers an open and unified framework for caching. By against, traditional caching systems are application-dependent and caches use proprietary protocols. Even if some studies have tried to build a transparent cache, this issue remains harsh and expensive to achieve [Alimi 2013]. In addition, previous caching technologies are designed for a specific traffic type, however, caching with ICN is expected to deal with several types of traffic. There is another difference which concerns

the network topology. In fact, a cache element in traditional caching systems is usually located in a well-known predetermined location, whereas, in ICN, every node can potentially be a cache node for any content [Rosensweig 2010]. To recapitulate, in-network caching in ICN, contrarily to traditional caching, may be low-complex, topology-aware and adaptive to different and dynamic Internet traffic.

Despite the plenty existing studies that support traditional caching [Dahlin 1994, Michel 1998, Fan 2000], in-network caching in ICN outperforms them [Katsaros 2011, Dan 2011, Wang 2013a]. Indeed, Pavlou, in his keynote speech [Pavlou 2013], quotes that the deployment of the ICN in-network caching exhibits significant improvement in terms of less pressure on bandwidth and better delivery efficiency. However, in-network caching can present some decision challenges: the cache placement, (where to cache), content replacement (which content is to evict from the cache) and request routing (how to redirect requests to an optimal cache). In this thesis, we address, in the context of in-network caching, the cache placement and content replacement challenges. We start by studying the cache placement and we present, in this section, the in-network caching in ICN IoT networks.

Unlike traditional Internet hosts, IoT devices are resource-constrained in terms of memory, energy and processing power. In addition, IoT data are usually small and frequently transient compared to the Internet contents which are large and time-invariant [Minerva 2015]. Furthermore, IoT equipment can be mobile. IoT applications impose stringent requirements on some critical domain application as eHealth. They focus on the freshest data. Facing these distinct particularities of IoT systems, some researches have addressed the in-network caching challenge in ICN/IoT networks.

The work of [Vural 2014] is the first study addressing the in-network caching of IoT data at content routers on the Internet. Authors of this work discussed the trade-off between the cost of multi-hop communication and the freshness of a requested data. In this context, they introduced a distributed probabilistic caching strategy in which cache nodes dynamically update their caching probability. This probability depends on the distance between the producer and the consumer as well as on the data freshness in a way that the closer the caching location is to the producer, the fresher the retrieved *Data* packet is. Two results were observed. First, the more popular the content is, the higher the cost reduction will be. Second, when caching nodes are closer to consumers, IoT data provide a lower workload on the network.

As discussed in [Quevedo 2014a], using the already existing ICN caching mechanisms leads to a considerable reduction in the consumed energy and bandwidth. Authors demonstrated that as the available caching storage capacity increases, the consumption

of both energy and bandwidth decreases. They affirmed that due to the fact that IP-based designs are becoming more complex and hard to achieve, the deployment of ICN in several important fields makes a positive contribution.

Hail et al. in [Hail 2015] proposed a novel distributed probabilistic caching strategy named probabilistic CAching STrategy for the INternet of thinGs (pCASTING) that considers data freshness and potentially constrained capabilities of IoT devices. Among several ICN instantiations, they refer to the NDN architecture since it leverages simple and robust transmissions of requests and contents. Results show the effectiveness of the proposed scheme in terms of energy consumption and content retrieval delays.

Vural et al. studied in [Vural 2016] the in-network caching of transient data in IoT systems. This paper treats the cache freshness in ICN-IoT scenarios. Authors present an analytical model that captures the trade-off between the communication cost and the data freshness. They aim to quantify caching gains with transient items. Through simulations, they verify caching benefits and demonstrate that caching transient data is entirely possible.

Despite the efforts of the researchers to deal with IoT applications' requirements, obtained results are not sufficiently mature. In fact, concerning the IoT freshness, existing solutions use a fixed or probabilistic lifetime for cached data. However, sensors events are unpredictable. For this reason, we specifically focus, in the next section, on cache freshness challenge.

3.4 Cache freshness

Cache freshness/coherence maintains the validity of the shared contents stored in multiple caches. As an immediate consequence, consumers can trust copies in caches. A copy is considered valid when it has the same version as the source, or when the difference between the data in the source and its copy in the cache is not important. For example, for a home automation usage, a cached temperature value of 23° is considered valid when the ambient temperature is equal to 24°.

In [Dingle 1996], Dingle et al. have focused on the web cache coherence. The problem of cache freshness on the Web or on ICN remains the same. This issue is also addressed in the area of distributed systems. Maintaining freshness is easier in web caches than in distributed files because these latter support distributed writes. Whereas, updating web content is performed only by the content's producer. In the sequel, we introduce the representative freshness mechanisms used by popular distributed file systems. These mechanisms can be classified into two major classes. First, the Valida-

tion check [Sandberg 1985] mechanism in which the validity of the requested content is checked with the time-stamp (caching time). The time-stamp is sent to the producer to check if the data has not been modified since it was cached. Second, the Callback mechanism [Kazar 1988], in which all cache nodes are notified by the producer whenever an update occurs. So, the producer must have the list of all content's caches, which is unsuitable for constrained-devices.

Concerning the web-freshness mechanisms, there are four major approaches. First, the HTTP Headers mechanism [W3C 2006], in which an If-Modified-Since header is appended to the request in order to check if the last modification time of the content is greater than the If-Modified-Since time in the request header. This is called a conditional Get. Second, the "Naive" coherence [W3C 2006] which is proposed for hierarchical topology. This mechanism is a particular case of HTTP headers. In fact, to check the content validity in a cache, the latter sends a conditional GET message only to the next higher cache or server. Third, the Expiration-based freshness mechanism [W3C 1996]. In this one, each content is marked with an expiration time. The content is assumed valid as long as expiration time has not elapsed. Fourth, pre-fetching mechanism [Berners-Lee 1995] consists in periodically refreshing the cached contents. This is performed by sending conditional Get requests to check if the content is changed. The main disadvantage of this mechanism is the decision of contents refreshment time. This decision depends on the popularity of a content and its age. In some cases, it may be useless to refresh it.

Authors in [Quevedo 2014b] address the cache freshness issue in ICN. They proposed a freshness parameter. Unlike the defined parameter in [Hail 2015], the suggested one is not constant. It can be adjusted by consumers to specify their particular freshness requirements. Even if simulation results show that the received data validity has been improved, this solution remains inefficient in case of a scenario with very transient data. This is because consumers expectations can not avoid receiving outdated data.

The cache freshness is an important challenge that threatens the in-network caching advantages, especially with a high-dynamic environment in term of data updates. This issue, despite its importance, is still in its infancy. The cache coherence will be treated in this thesis.

Another point that can influence the in-network caching performance is the cache replacement. Since the cache size is limited, once the cache is full, some stored content must be deleted to allow caching new items. To this end, cache replacement policies are used to choose the data object to be evicted. We present, in the following section, proposed cache replacement policies.

3.5 Cache replacement policies

Limited cache sizes require the use of content replacement policies. This can affect the cache efficiency. In fact, one of the objectives of the in-network caching was the requirement for low content delivery delays. So, if finally cached contents are evicted without satisfying any request during their lifetime in a cache, this cache is inefficient. As a consequence, in-network caching efficiency depends not only on the caching strategies but the replacement policies as well. In addition, in some domains, we highlight the need for low overhead implementations.

Traditional cache replacement policies were designed for computer architectures and databases [Chankhunthod 1996]. The dominant cache replacement policy is First In First Out (FIFO). FIFO is the default caching policy in NDN [Chiocchetti 2013b]. In this caching policy, the oldest data object is replaced by the newly arriving content. The FIFO policy has a slightly simpler implementation in comparison to the other strategies because it follows the same policy to store contents in the cache.

With the appearance of P2P and CDN, cache replacement policies have begun to be used in web caches [Podlipnig 2003, Wong 2006]. Existing policies in the web can be classified into four categories; the recency-based policies, the frequency-based policies, the size-based policies and the randomized policies. The rationale behind the recency-based category is that recently requested contents are more likely to be requested again in the near future. These policies are more efficient in the case of high temporal locality of request streams. This means that there is an important number of consumers interested in a common set of data objects. The most known policy in this category is the Least Recently Used (LRU). LRU has been adopted in a considerable number of caching policies [Psaras 2012, Cho 2012, Sourlas 2014]. This eviction policy replaces the data object that is not being used for the longest time. LRU is efficient for line speed operations because both search and replacement tasks can be performed in constant time (O(1)).

Frequency-based policies are based on objects' popularity. The rationale behind this category is to keep popular objects in the cache to satisfy a high number of requests. For instance, this category is efficient with web pages that provide news or new films. The least frequently used (LFU) policy is the simplest variant in this category. With LFU, the cache node keeps track of the number of times a data object satisfies a request. LFU purges the item with the lowest content frequency. However, unlike LRU, the implementation of LFU is computationally expensive since it cannot be implemented in such a way that both search and replacement tasks can be executed in constant time.

Table 3.1: Cache replacement policies

Category	Description	Existing Policies	Representative Policy
Recency-based	Keeps the recently referenced objects	LRU, LRU-threshold, LRU*, LRU-hot, SLRU, HLRU, LRU-LSC, SB-LRU	LRU
Frequency-based	Keeps popular contents	LFU, LFU-Aging, LFU-DA, swLFU, Window-LFU	LFU
Randomized policy	A simple random choice to avoid high computation overhead	RR, RAND, HARMONIC	RR
Size-based	Evicts large contents	Size, PSS, CSS	Size

LFU is sometimes combined with LRU and called LRFU. The drawback of LFU policy is that popular contents which become unpopular remain in the cache for a long time.

The randomized policies are specifically designed for cache nodes with complex data structures. The RR policy, the simplest randomized policy, is a Random Replacement policy which replaces a randomly chosen cached item. Obviously, this strategy is not complex in terms of search and replacement tasks.

Finally, the size-based category depends on object size as the primary factor. This category comes with the logic to evict large object in order to provide room for many small objects. This category works well with web sites that maintain more text file than multimedia. The basic policy in this category is Size.

We conclude that FIFO, LRU, RR and Size replacement policies can perform both operations in constant time, the complexity of LFU replacement operation is O(c) and grows linearly with the cache size c.

We summarize in Table. 3.1 different categories of cache replacement.

LRU, LFU, FIFO, RR and Size are already used on the web since a long time. Some other strategies are recently proposed in the context of ICN. In [Panigrahi 2014], authors proposed a Universal Caching (UC) strategy designed for ICN. UC makes the replacement decision depending on a parameter assigned to any incoming content. This parameter includes the distance from the source, reachability of the router and the frequency of the content access. Al-Turjman et al. in [Al-Turjman 2013] introduced the Least-Value First (LVF) cache replacement policy which takes into account the delay for retrieving content as well as the popularity and age of content experienced by a node.

It is worth noting that both content replacement policies and caching strategies are

important to evaluate the in-network caching performance and may complement each other. We address in the following section the mobility aspects of IoT devices.

3.6 Mobility in ICN

In ICN data objects are requested without any location information. Therefore, mobile devices are no longer reachable during and after the handoff. The problem is more serious in IoT scenarios, because, first, due to the advancement in IoT technology, the mobility frequency is very important such as sensors deployed in eHealth applications and intelligent transport systems. Second, IoT applications require keeping data continuity, especially with real-time sessions.

To support the mobility in ICN, it's important to specify the network architecture and the packet transmissions. Since we have proved in chapter 2 that NDN architecture is the most convenient architecture for IoT environment, we are interested, in the remainder of this chapter, in the mobility challenge in NDN.

3.6.1 Mobility management in NDN

The mobility in NDN can be classified into *Consumer* mobility and *Producer* mobility [Zhou 2014]. The former can be easily handled since NDN is a receiver-driven approach. In fact, when a consumer moves, it just needs to resend unsatisfied *Interest* packets from the new location. As shown in the IoT scenario in Figure. 3.1, the *consumer* moves from router *r2* to router *r1*. It just sends an *Interest* from *r1*. According to FIB tables, the request is routed by name through *path3* rather than *path1* and the requested data will be satisfied using the same request path. The only bad consequence resulting from the consumer mobility is the loss of *Data* packets. This occurs when the consumer moves after sending an *Interest*.

Concerning the producer mobility, it is more complex. This complexity is due to the fact that in NDN, naming scheme does not include any location information. In addition, routing the requests only depends on FIB entries. As a consequence, producer mobility can cause the loss of *Interest* packets. In Figure. 3.1, the *consumer* has sent an *Interest* packet along *path1* to retrieve the content *Person/health/Blood/Pressure*. When the producer maintaining this content moves from router *r7* to *r6*, the *consumer* can not be able to reach this content because the router *r5* will redirect the request to *r7*. To satisfy the request, the *Interest* packet should follow *path2*. To overcome this problem, the FIBs should be updated with the new path to the producer. So, the router *r5* will match the entry *Person/health/Blood/Pressure* in its FIB with the right router.

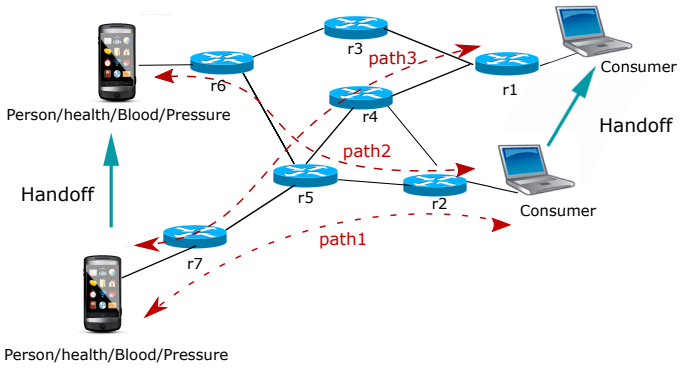


Figure 3.1: Mobility management in NDN/IoT

NDN utilizes the flooding solution to support the producer mobility. After a handover, the producer broadcasts its new path in order to update all the FIBs in the network. This solution results in a heavy bandwidth cost and a large handover delay. For this reason, some studies were interested in the producer mobility issue in NDN and have proposed some solutions to deal with this problem. We review some studies in the next subsection.

3.6.2 Proposals for producer mobility issue in NDN

In this subsection, we present existing producer mobility solutions in NDN. They can be classified into four approaches; location resolution approach, triangular approach, locator/identifier separation approach and routing-based approach.

3.6.2.1 Location Resolution approach

Mobility in this approach is handled by a home agent like entity that acts as relays of *Interest* packets. This category, in general, requires a Location Resolution System (LRS) to bind information between the name prefix of a content producer and its current location.

The Figure. 3.2 shows the operation of this approach. When the producer changes its location (arrow 3), it updates the entries of the contents attached to it in the LRS (arrows 4-5). We assume that the producer name is *Person/health/*, then, all the entries with this prefix will be updated. The LRS contains mapping information between the content name *Person/health/Blood/Pressure* and the router */home/*. The *consumer* asks first for this location (arrows 6-7) then sends the *Interest* with this location information (arrows 8-9).

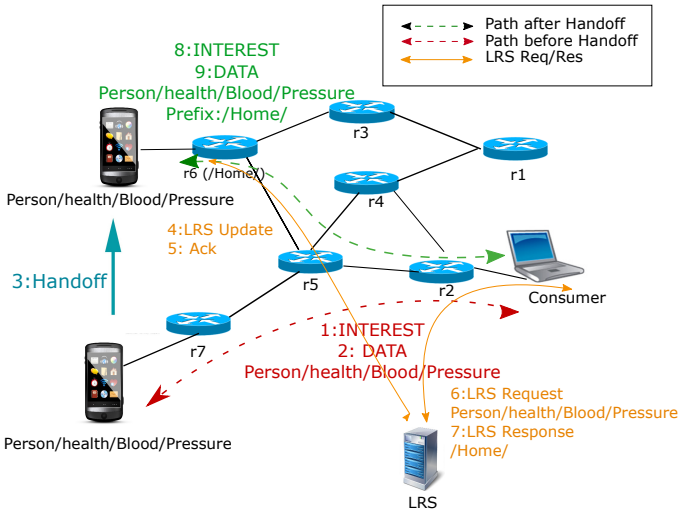


Figure 3.2: Location Resolution approach

Studies in [Lee 2012] and [Hermans 2011] relies on this concept. They use a specific router as a tunnel which is called respectively home router and indirection point. Proposals in [Kim 2012] and [Zhu 2013] also use a third entity which is a server to store the current location of the content provider. Unlike the two first proposals, in [Kim 2012] and [Zhu 2013] the prefix is set as a metadata in the *Interest* rather than the name itself.

3.6.2.2 Triangular approach

The general principle of this approach is that *Interest* packets are routed by name normally to the old position, then they are redirected to the new one. This method uses the FIB update mechanism to establish the redirection path. The approach operation is shown in Figure. 3.3. When the producer moves to a new location (arrow 3), it sends a notification in order to update FIBs in the intermediate routers (r5, r6 and r7) between the old and the new location (arrows 4-5). The *Interest* packet sent by the consumer will follow routers r2, r5, r7, r5 then r6 (arrows 6-7).

Proposals in [Kim 2015, Han 2014, Wang 2013b] are based on this approach. The first one [Kim 2015] uses the *Interest* packet to update FIB tables, however the second [Han 2014] uses the *Data* packet. Concerning the third proposal [Wang 2013b], it implements the greedy routing to update the intermediate FIB tables.

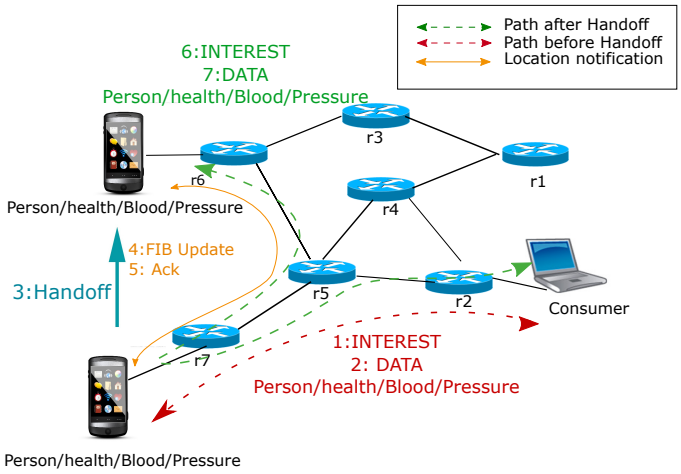


Figure 3.3: Triangular approach

3.6.2.3 Locator/identifier separation approach

In this approach data objects have two names; an identifier which is used to send *Interest* packets and a locator which is used in the routing protocol. The locator/identifier mapping is done in a home router, then the locator is used to route the *Interest* packet to the right router.

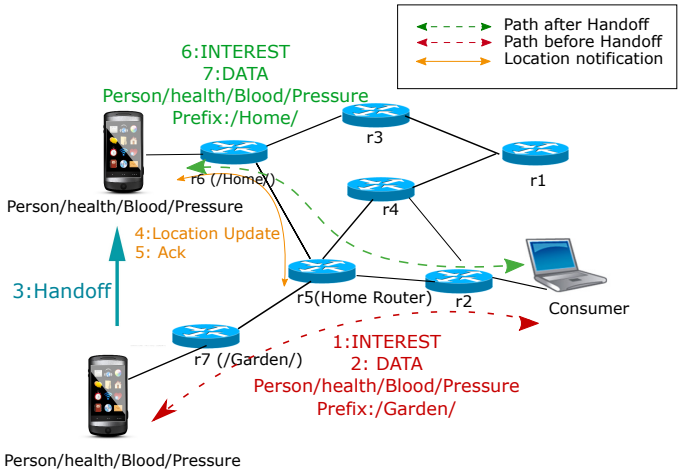


Figure 3.4: Locator/identifier separation approach

The Figure. 3.4 shows the operation of the locator/identifier separation approach. The content object managed by the producer has as an identifier *Person/health/Blood /Pressure* and a locator */Garden/*. When the producer changes location (arrow 3), the identifier remains the same, however, the locator becomes */Home/*. So, it notifies the

home router about this mobility, in order to update its FIB (arrows 4-5). When *Interest* packets pass through the home router, the locator is added and these modified *Interest* packets are routed to the current location based on added information (arrows 6-7).

Studies in [Ravindran 2012, Hermans 2012, Rao 2014] are locator/identifier separation-based. Two ways are proposed in this approach. The first way is to add an optional location named field as in [Ravindran 2012]. The second way is that the name contains both location and identifier as a prefix, for instance. Authors in [Rao 2014, Hermans 2012] introduce schemes that use this way.

3.6.2.4 Routing-based approach

Proposals based on this approach admit that a dynamic routing approach can solve the producer mobility problem. The concept of this category is to be able to reach the data object using a good name-based routing strategy. Figure. 3.5 shows that it is easy to retrieve the requested data (arrows 4-5) after handover. However, the complexity of this approach depends strictly on the routing algorithm. In fact, a name-based routing protocol should be able to discover temporary copies that may be cached outside the ordinary paths towards original content producers.

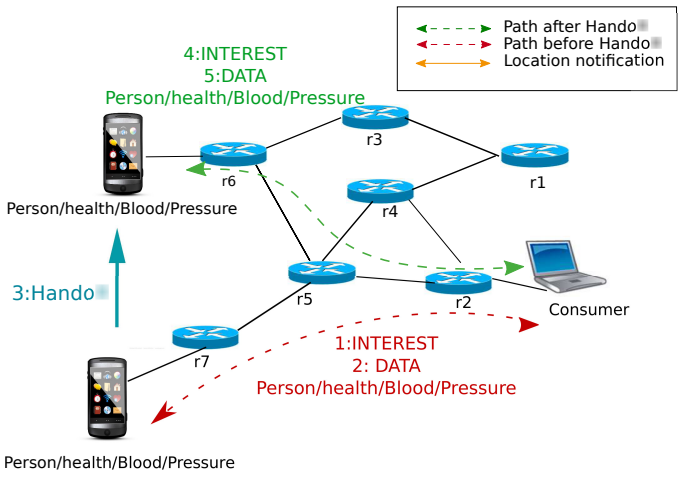


Figure 3.5: Routing-based approach

Authors in [Wang 2013b, Azgin 2014] introduce routing-based mobility management. In [Wang 2013b] authors propose a routing protocol that considers distances from the destination to the current router neighbors which is called Greedy routing protocol. Concerning the proposed architecture in [Azgin 2014], it leverages a decentralized routing strategy based on flooding.

3.7 Conclusion

The concept of ICN has been the object of several researches in the last few years. This innovation coincided with the evolution of IoT environments in terms of the increased amount of IoT devices and IoT traffic in the network, the heterogeneity of devices and communication technologies, and the mobility of *Things*. Some studies argue that ICN is a promising candidate for IoT systems. In this chapter, we cited studies that supported this challenge. We especially focused on in-network caching and we presented existing caching studies. Furthermore, we detailed cache replacement policies. Facing the particularities of IoT systems in terms of data transient and mobility, we were interested in studies that target cache freshness and mobility management. In the following part, we will present our contributions within the in-network caching and the mobility support in ICN/IoT systems.

Part II

Contributions and Results

Chapter 4

Caching strategy for NDN-based IoT networks

Contents

4.1 Introduction

IoT networks originate traffic patterns very different from the popular Internet applications. In addition, they require specific procedures as discovery and management. Concomitantly, unlike high-performing Internet routers, IoT devices are mainly resource-constrained nodes. As a consequence, packet transmission performance may be very poor. Thereby, NDN modules must be optimized in order to support such limitations.

The in-network caching is one of the pillars of ICN which can significantly improve the performances of IoT networks. It increases the data availability in the network since consumer's requests are most of the time satisfied by cache nodes rather than the producer. Thanks to this feature, the traffic load is significantly reduced. In addition,

caching affords the reduction of the required distance for data retrieval and as a consequence, the response latency diminishes. Furthermore, it avoids bottleneck caused by publishing data at a unique location.

Before going into the details of our contributions, we need first to understand the IoT architecture. Until now, there is not yet a consensus on the physical model of the IoT architecture to adopt. Two divergent visions are tacitly used in various experiments of the IoT. The first considers the IoT as a collection of sensors/actuators interacting with each other without any intermediate equipment. The second vision assumes that actuators and sensors communicate through their respective gateways. In an IoT scenario where actuators and sensors directly communicate with each other, an actuator directly requests a sensor to get the measured value by the latter. This scenario assumes that the actuators and sensors are provided with an operating system, software applications and memory. In the second model, gateways play an important role. In fact, actuators, as well as sensors, are directly connected thereto. Contrary to the first view, the measured value by a sensor is no longer stored in the memory of the latter but in the gateway to which it is attached. Starting from the premise that IoT devices are resource-constrained, caching data, as small as they are, in a sensor memory, will be the determinant of embedded applications and operating systems. In addition, the caching decision is made by the CPU of the sensor which decreases the battery lifetime. For these reasons, we propose to adopt, in this work, an NDN-based IoT architecture where data are cached on the gateways and sensors represent the producers.

In this chapter, we aim to address the caching issue and we propose a caching strategy which considers IoT ecosystem limitations. To this end, we detail caching techniques, then we fix some caching strategy assumptions to propose a caching strategy suitable for IoT environment. To evaluate our proposal, we finally analyze the cache cost. Just prior to the caching study, we start this chapter by giving an overview of the NDN architecture in order to position our contributions and precise the modules on which we will act.

4.2 A focus on NDN layer

We portray in Figure. 4.1 the NDN architecture overview. In the under layer, we found the *Thing layer* which includes a multitude of devices of the IoT ecosystem. IoT equipment are heterogeneous and also equipped with heterogeneous communication interfaces. In addition, they exhibit different mobility patterns and constraints (memory, battery, processing power). Concerning the IoT applications, they are represented at the top of

the layers imposing requirements that should be satisfied. To cope with this challenge, NDN layer is positioned in the middle and acts as a networking layer. This latter expects to hide to *Application layer* the complexity and diversity of the underlying things by adapting its modules to their features [Amadeo 2014a]. NDN modules are depicted in the *Core NDN Protocol* offering two main services which are *Caching* and *Routing*. The former includes *Caching strategies*, which is addressed in this chapter, and *Cache Replacement Policies* module which is detailed in chapter 5. The *Routing* module englobes both the *Data* and the *Interest* packets routing. This module will be presented in chapter 6. As we have mentioned in chapter 2, an NDN node is composed of three structures; CS, PIT and FIB. We precise in Figure. 4.1 that only the CS structure participates in the Caching module and all of them participate in the Routing module. In fact, each step of the *Interest* forwarding process (red arrows) includes lookups to CS, PIT and FIB. The *Data* forwarding is done according to PIT entries and requested *Data* can be stored in the CS based on the adopted caching techniques (green arrows).

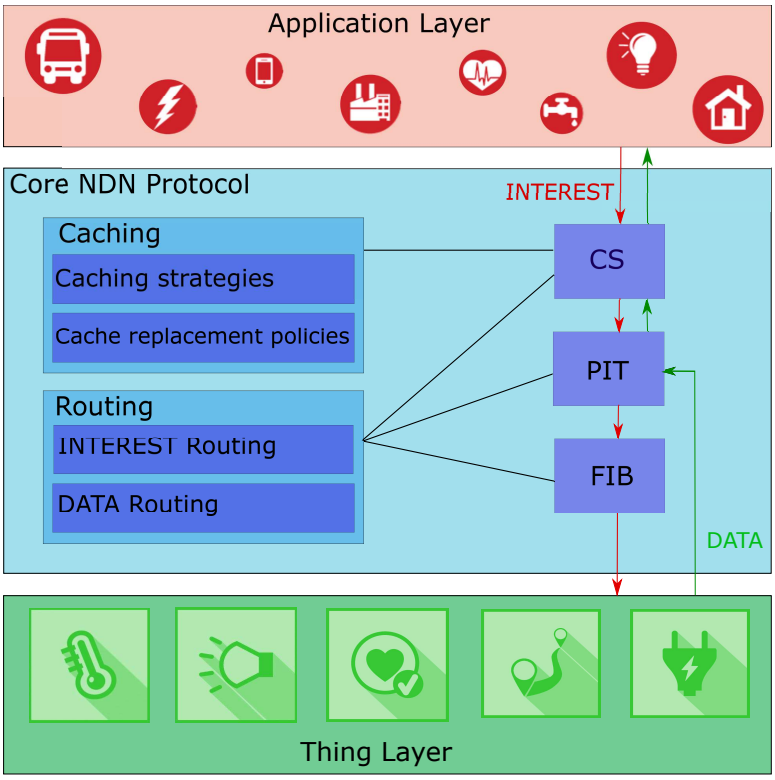


Figure 4.1: NDN Layer

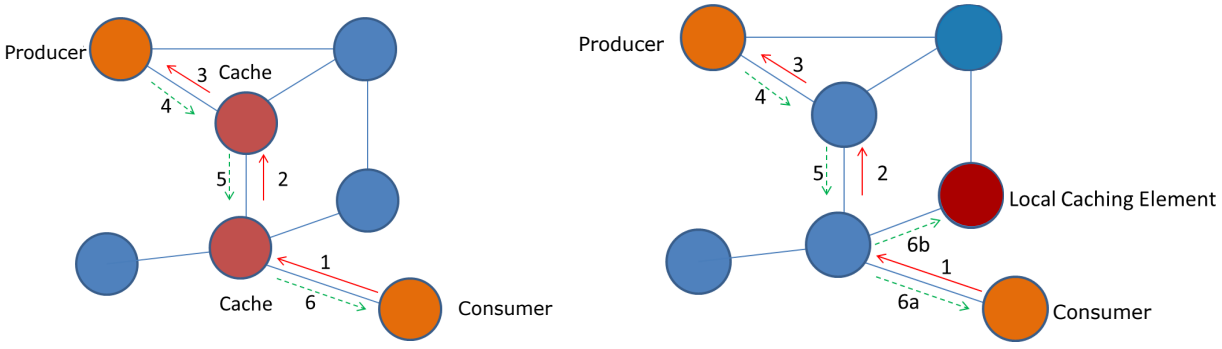
4.3 Caching techniques

The in-network caching is a matter of choosing the location of the different copies of a given content and adapting it according to the geographical arrangement of the clients and the nature of their requests. The caching is performed in two phases; the first phase is the selection of nodes that will participate in the caching process which we call *cache placement selection*. The second phase is the *cache decision* in which each cache node decides to cache or not the content.

4.3.1 Cache placement selection

The cache placement selection is classified into two scenarios: *On-path* and *Off-path* caching [Xylomenos 2012]. We detail in Figure. 4.2 the operation of the two scenarios and we represent selected cache nodes by red nodes. With the On-path scenario, cache nodes belong to the request path. In fact, as presented in Figure. 4.2a, when the requested content is forwarded to the consumer following the request path (arrow 4-6), nodes residing in the path choose either to cache the content or not according to their specific cache decision policy.

By applying the Off-path caching scenario, cache nodes are fixed in the topology. In fact, each consumer has one associated cache element known in advance as shown in Figure. 4.2b. When the producer will send the response, it multicasts the content to both the consumer (arrows 4-5-6a) and the associated cache (arrows 4-5-6b).



(a) On-path Caching (b) Off-path Caching

Figure 4.2: Cache placement

From Figure. 4.2, we can trivially remark that, unlike on-path caching, off-path caching requires additional overhead. Furthermore, the off-path caching only allows serving local consumers by off-path caching points. Off-path caching is very hard to

achieve on the Internet since it requires coordination between many application-level caches.

4.3.2 Cache decision policies

The cache decision policy allows choosing for each cache node either to cache the content or not. The decision is taken for each response and can change from a request to another. Concerning the off-path caching scenario, the selected cache node depends on the consumer. The decision is then always positive. However, with the on-path caching scenario, a cache decision policy is essential. Generally, to optimize the use of the cache, the participating nodes cooperate with each other. Information about the popularity of the content will be exchanged to influence the choice of location. Several caching strategies have been defined. They are classified according to the nature of the coordination between the participants in two categories: caching with explicit coordination and caching with implicit coordination [Zhang 2013]. With the explicit coordination approach, the nodes exchange information that describes the state of the different local caches, such as the size occupied, the request frequency of the different objects and the distances between replications. Then, each participant takes this information into account when making decisions. This allows optimizing the overall performance of the entire architecture. According to nodes participating in this decision, we can define three levels of explicit coordination; global coordination, path coordination or neighborhood coordination. In the global coordination, the placement decision involves all cache nodes, while in the path and neighborhood coordination nodes used are respectively nodes belonging to the request path and cache's neighbors nodes. The explicit coordination requires a periodic exchange of information between nodes to keep a consistent state. This reduces the performance of this approach.

In implicit coordination, each participant node individually decides whether to cache the data or not, without being informed of the decision of the others. This approach avoids the exchange of information in the caching process which minimizes the bandwidth consumption. However, by neglecting the decision of the other nodes, redundant and near copies are possible.

Initially, in the ICNs, only one implicit cache decision policy is adopted. It is the Leave Copy Everywhere (LCE) approach [Laoutaris 2004, Jacobson 2009, Zhang 2010]. According to this approach, the participating nodes systematically copy each received data into their local space. This simple approach does not require any exchange between the nodes. Furthermore, it generates a high redundancy rate in the infrastructure, providing fast delivery and low bandwidth consumption. Several new stud-

ies have shown that this approach is costly and infeasible and also inefficient to account for the exponential growth in the amount of data exchanged over the Internet [Kutscher 2011, Rossini 2012, Chai 2012, Fricker 2012, Rossi 2012]. Thus, this result has prompted the research community to deepen their research to propose specific ICN cache techniques.

We present in Figure. 4.3 the widely used caching strategies. In this figure, green nodes represent caches.

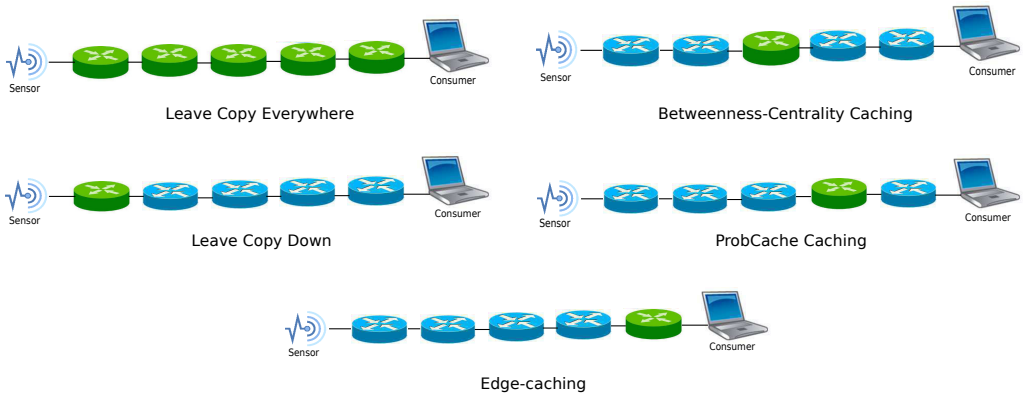


Figure 4.3: In-network caching strategies

The Leave Copy Down (LCD) strategy [Laoutaris 2006] leaves a copy in just one node which is the gateway on one level down in the reverse path towards the consumer. The betweenness-centrality strategy (Btw) [Chai 2012] depends on the betweenness-centrality parameter which is calculated for each node in the topology. It measures the number of times that a node belongs to a path between any two nodes in the topology. The node with the highest betweenness-centrality parameter stores a copy of the data. The Probabilistic Cache (ProbCache) caching strategy [Psaras 2012] decides to cache a copy with a probability inversely proportional to the distance between the consumer and the producer. As a consequence, this strategy privileges caching close to consumers. The edge-caching (Edge) strategy [Fayazbakhsh 2013] is mainly conceived for hierarchical topology. It also caches data in one node as with LCD, however, with this strategy, the position of cache node is totally the opposite, it is the last gateway in the reverse path towards the consumer.

We note that strategies presented above are not complex and do not need costly calculations. Some other studies have proposed more complex in-network caching strategies for ICN with the aim to increase the system performances in term of response latency, the producer load and the required distance to satisfy a request. These advanced researches

target more outcomes such caching redundancy [Wang 2013a, Psaras 2014], network congestion [Badov 2014], supporting diverse traffic types [Tsilopoulos 2011], routing of requests to the nearest cache [Eum 2012]. It is worth-noticing that in-network caching efficiency can be strictly dependent on the context in which it is used. In fact, each context has its specificity in terms of traffic volume and type, the frequency of content generation, consumers' expectations and the ability of machines to support heavy computation.

We assume that the location at which the caches are placed play a prime role in the resulting traffic and load reduction. Thus, addressing the location problem of caches is an important part of the campaign for in-network caching. We remind that, in this thesis, we consider IoT environments. We then propose an IoT context-aware caching strategy and we start by giving the assumptions of our proposal in the next section.

4.4 Caching strategy assumptions

In an IoT context, we believe that it is preferable to distribute caches on the network topology (on-path) rather than centralizing them on fixed caching nodes (off-path). In fact, with an important number of devices and large data flows, specifying particular gateways to handle caching (even locally) would have as an immediate consequence the creation of multiple congestions within the network. In addition, the off-path caching increases the signaling overhead since it requires, in the caching process, additional packet transmissions to the cache node. On the other hand, being in the case of the on-path caching scenario, we admit that the implicit coordination is the most suitable cache decision approach for IoT networks. In fact, being aware of the tremendous growth of the IoT traffic, prompted us to avoid as much as possible extra packet transmission. To recapitulate, the first assumption assumes to follow an on-path cache placement selection combined with an implicit cache decision policy.

W.K.Chai et al. showed in [Chai 2012] that caching in fewer nodes can perform better results in terms of producer hits and number of the traversed hops to reach a content. Even caching in one randomly selected cache can be better than the LCE strategy. In fact, with LCE, caches fill up very quickly because the cache size is very limited compared to the number of the contents to be cached. Once the cache is full, a replacement process is triggered to allow caching new items. The more frequent the replacement process is the least efficient the caching is, because contents are cached then deleted without being used and it just increases the caching cost. In addition, IoT nodes are memory and energy-constrained, consequently, it is more advantageous to minimize

the resource consumption. As a consequence, the second directive aims to reduce the number of caches. It should be emphasized that this does not mean that lowering the number of cache nodes provides a better strategy. Rather, it is a trade-off between the number of evictions, the cache cost and the data availability.

From this point of view, we need to know what is the selection criterion of the placement of cache nodes in a topology. Authors in [Fayazbakhsh 2013, Dabirmoghaddam 2014, Imbrenda 2014] prove that the most efficient nodes for implicit caching are located in the edges of the topology. This is due to the fact that generally consumers are connected to edges, so cached copies will be easily reachable. Recall here that the edge-caching strategy is originally conceived for hierarchical topologies, under this strategy, caches are located at the leaves of the topology. Since, in IoT networks, consumers are generally located at network edges and in other cases in the middle of paths, we aim to improve in-network caching benefits and we propose as the third assumption to cache at nodes close to consumers.

4.5 Consumer-cache caching strategy

Based on the three assumptions presented in the previous section, our caching strategy, termed *consumer-cache*, proposes to store a copy of contents on the on-path nodes which are connected to a consumer. We can say that if all nodes in the topology are connected to consumers, consumer-cache strategy tends towards LCE strategy. Moreover, if consumers are only connected to edges, our strategy will be similar to edge-caching strategy. Our strategy strictly depends on the number of consumers and their locations. We present below the algorithm of the consumer-cache strategy (Algorithm 1). It is assumed that the request has already arrived at the producer, which will send the desired content to the consumer.

As we can show in the presented algorithm, the decision to cache or not the content is checked at each node v_i from the producer to the consumer (Line 16). When v_i is connected to a consumer (Line 17), this cache node is added to $Cache\_nodes$ list (Line 18). Then, we verify if the *Data* is already stored in this cache (Line 19). If it is the case, just an update of the existing item is occurred (Line 20), else the content is added in the CS structure (Line 22).

Let us take the example of the topology presented in Figure. 4.4 where $Consumer1$ requests a content named (c). Initially, all caches are empty. So, the request is forwarded to the producer. $Producer1$ holds the content (c). The latter sends the response via the path from $n11$ to $n0$. While forwarding the response, each of the nodes in the response

Algorithm 1 Consumer-cache strategy

Input: $Data$ to cache
Output: The set of selected $nodes$ as a cache
Data:
$Prod$ = the $producer$ node
$Cons$ = the $consumer$ node
$G = (V, E)$ a graph that represent the network
$Path = (v_1, v_2, ...v_n) \in V * V * ... * V$ s.a:
v_i adjacent to v_{i+1} for $1 \leq i \leq n$ and
$v_1 \longleftarrow Prod$
$v_n \longleftarrow Cons$
$Cache\_nodes = (v_c, ..., v_p) \in V * V * ... * V$ s.a:
$1 \leq c \leq p \leq n$
$Cache = (c_1, c_2, ...c_m)$ s.a: $m \leq cache\_size$
Begin
$i = 1$
while $i \leq n$ **do**
  if $\exists$ a consumer connected to node v_i **then**
    $Cache\_nodes \leftarrow v_i$
    if $DATA \in v_i.c_{1..m}$ **then**
      $c_i.value = DATA.value$
    else
      Cache($Data$,v_i)
  i= i+1

path has to decide whether to cache or not a copy of (c) according to its own caching strategy. With LCE, all the 10 nodes in the path store a copy. Other strategies require that the content should be stored in only a single node. For LCD, only $n11$ retains a copy; whereas with Btw, the cache is $n6$. However, under consumer-cache, caches are $n7$ and $n0$. Concerning the edge-caching, $n0$ is the selected cache node. Also with ProbCache, $n0$ has the greatest probability to be the cache. Let us now assume that $Consumer4$ requests the same content (c). It sends a request towards $Producer1$. With Btw and probably ProbCache, we have a cache miss. Under the ProbCache strategy, the probability to find the content in $n7$ is very low as this node is far from Consumer 1. Using consumer-cache and LCE the request is satisfied by $n7$. In the case of LCE, the cached copies at $n0$, $n3$, $n4$, $n5$, $n6$, $n8$, $n9$, $n10$ and $n11$ are redundant. It is therefore clear that caching the content close to consumers is sufficient to avoid redundancy in other caches.

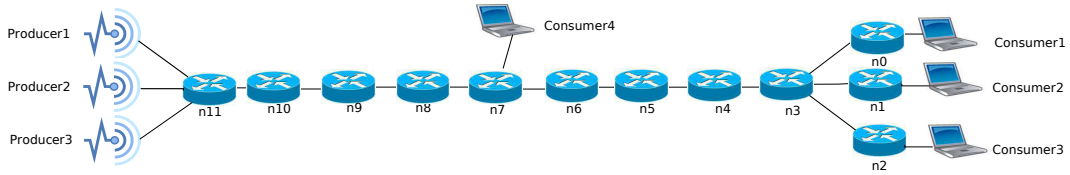


Figure 4.4: An example with a simple path

4.6 An example using the consumer-cache strategy under NDN/IoT networks

We describe in Figure. 4.5 a scenario under the Named Data IoT Networks using consumer-cache caching strategy. In this scenario, $Consumer1$ sends an *Interest* packet for the content named $/Home/room1/Tmp$ (arrows 1-5). When the node $n2$ receives the *Interest* message, it first performs a lookup in it CS. If there is no entry with this name, the request is forwarded to the next hop according to the FIB of this node. This base contains an entry which points to node $n3$. The router records the request's incoming interface $n1$ in the PIT. Then, it sends the *Interest* to $n3$ and so on. In the case of a cache miss, the producer satisfies the request. A *Data* message is then returned. This packet is forwarded based on states stored in PITs (arrows 6-10). Under the consumer-cache strategy, only $n3$ and $n1$ store the object in their CSs. Now, we suppose that $Consumer2$ asks for the same content $/Home/room1/Tmp$ (arrow 11). When the *Interest* packet arrives to node $n3$, it matches the information object found at the CS. Then, the *Data* is directly returned (arrow 12).

4.7 The cache cost

The *cache cost* is a compromise between data availability and caching overhead. It captures the trade-off between the average delay to receive a content and the corresponding caching cost [Vasilakos 2012]. The delay to obtain information is considered as a cost because, in the case of a cache miss, the delay to wait for the response by searching the caches is longer than the delay of directly looking for the content from the remote server. As we have already mentioned, the decision whether to cache a content or not in each node depends on the adopted caching strategy. So, the caching cost is related to this decision; the more we have "yes" decisions, the higher the global caching cost is. Although caching improves content availability, it increases redundancy and causes network overhead due to the existing content stored in the cache nodes. In addition, some

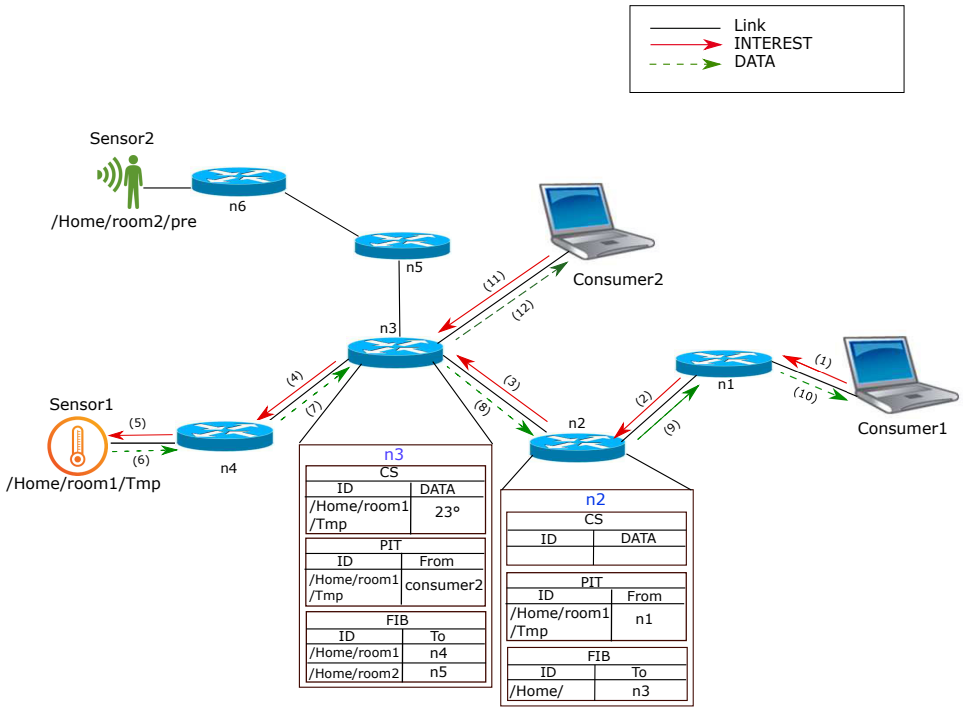


Figure 4.5: An example with the NDN architecture using the consumer-cache caching strategy

strategies, as LCE, come to fill up caches faster than other ones. In this case, we may remove still valid contents. As a consequence, the number of cache misses increases, and the cache is no longer effective. From this point of view, we consider the cache eviction as a cost; the more we evict contents, the higher the global caching cost is.

To calculate the cache cost, we were inspired by the cache cost function (Eq. 4.1) proposed in [Vasilakos 2012].

$$Totalcost = P_{hit} * C_{hit} + (1 - P_{hit}) * C_{miss} + N(S) * C_{cache} \tag{4.1}$$

Where P_{hit} is the probability to have a cache hit. It is calculated by the ratio of the number of requests satisfied by caches over the total number of requests. C_{hit} and C_{miss} are the average delays experienced by a consumer in order to receive its requested information from a cache and the producer, respectively. $N(S)$ is the number of caches in the topology. Finally, C_{cache} is the cost to cache items which is defined by the equation Eq. 4.2.

$$C^{j}_{cache} = \frac{a}{1 - \rho^{j}_{util}} \tag{4.2}$$

Where a is the cost coefficient that is set to 2 [Vasilakos 2012], and ρ^{j}_{util} is the cache utilization. The cache utilization at a node j is the ratio of the number of times a content is cached in j ($Ncaching$) over the number of times a content passes through j ($Ndecisions$). C_{cache} calculates the average of C^{j}_{cache} over j cache nodes. The proposed cost function (Eq. 4.1) captures the trade-off between the average delays for a consumer to receive an information item, given by the first and second terms of the equation Eq. 4.1, and the corresponding caching cost given by the third term of the equation Eq. 4.1.

To calculate the eviction cost, we should add, to the equation Eq. 4.1, a fourth term which is similar to the third one. However, the calculation of ρ_{util} differs from that calculated in the third term. In fact, in the cache eviction function, the utilization is the ratio of the number of times an eviction is performed in node j ($Nevictions$) over the number of time a content is cached in j ($Ncaching$). This is because the eviction takes place when a new item will be cached and the cache is full.

To summarize, we define our cache cost metric in the equation Eq. 4.3.

$$\begin{aligned} Totalcost = P_{hit} * C_{hit} + (1 - P_{hit}) * C_{miss} \\ + N(S) * \frac{2}{1 - \frac{Ncaching}{Ndecisions}} + N(S) * \frac{2}{1 - \frac{Nevictions}{Ncaching}} \end{aligned} \tag{4.3}$$

Collected values show that the sum of the third and fourth parts in the cost function which represents caching and eviction cost is notably higher than the sum of the first and the second parts which illustrate the delay. For this reason, we use normalization method to adjust the two values. The normalization method, named "reduced-centred", is calculated as shown in the equation Eq. 4.4, where μ is the average of the distribution, and σ represents the standard deviation of the distribution.

$$Z = \frac{X - \mu}{\sigma} \tag{4.4}$$

The findings of cache cost using different caching strategies will be given and analyzed in the last chapter.

4.8 Conclusion

In this chapter, we addressed an important feature of the ICN paradigm which is the in-network caching. We started by giving different caching techniques, notably the cache placement selection and cache decision policies. In order to propose a new caching strategy, we fixed three assumptions taking into account the specificities of the IoT environments. Then, we detailed our consumer-cache caching strategy and we gave an example under NDN-based IoT networks. The principal aim of this proposal is to deal with the constraints of IoT devices, for this reason, we analyzed the cache cost. In the next chapter, we will focus on the freshness requirement in IoT systems and we will present our two contributions; a cache replacement policy and a freshness-check mechanism.

Chapter 5

Freshness-aware in-network caching in NDN-based IoT networks

Contents

5.1 Introduction

In-network caching is a pertinent feature in NDN. In fact, satisfying requests by cache nodes can significantly improve the network performance in terms of response latency and producer load. However, to take full advantage of caching benefits, cache nodes should be efficient. The term efficient refers to two points. First, an efficient cache is a cache in which each stored content is useful by the consumers, so the data is not stored

then evicted without satisfying any request. Second, an efficient cache should provide consumers with valid and not out of date contents.

In an IoT context, data are transient and frequently updated by the producer. As a consequence, copies stored in caching nodes may become out of date after a certain period of time. Facing this stringent freshness requirement, we aim to consider the data freshness to privilege the eviction of stale contents from the cache and to check the validity of retrieved contents.

We address, in this chapter, this issue and we propose two contributions. We introduce a cache replacement policy named *Least Fresh First* (LFF) to manage the data eviction while considering the nature of the IoT environments. In fact, it is better to eliminate the invalid contents from the cache since anyway they are unusable. Then, we present a freshness mechanism called *event-based freshness* mechanism that checks the data validity before its retrieval from a cache node. In order to solve the freshness issue, we are based in the two contributions on a prediction mechanism according to the IoT traffic patterns.

This chapter is organized as follows, we first detail our prediction model, then we introduce the LFF cache replacement policy and the event-based freshness mechanism.

5.2 Analyze and prediction model

To identify non-fresh cached contents, we are based on the calculation of an expiration time for each item in the cache. In fact, a cached content is supposed to be fresh when the data has not been updated in the source after the last data retrieval. The basic idea of our approach is to calculate for how long the retrieved content will remain valid. We give the name T_{fresh} to this parameter. In order to calculate T_{fresh}, it is essential to study the sensors behavior to know the time of the next event of a sensor. To this end, we analyze and classify the flows of data generated by the sensors. We present, in the following subsection, the IoT traffic patterns.

5.2.1 The IoT traffic patterns

IoT traffic is usually classified into four categories: continuous, periodic, OnOff and request-response transmissions [Liu 2011]. With the continuous transmission, the data are transmitted in a continuous manner like video streaming. Under periodic transmission, the source sends a data at every fixed period of time T. For example, with a temperature sensor, after the elapse of each period (e.g: 1 hour), a new value is recorded. The OnOff transmission mode stipulates that the content is updated as soon as a new

event occurs. Let us consider the example of a presence sensor. The value 0 of the sensor indicates the absence of persons. Once someone is in the room, the value is updated to 1. Finally, in the request-response transmission, as its name indicates, the consumer sends directly a request to get the current value of the sensor. By considering these transmission modes, we can assume that sensors can be passive and do not follow any behavior as it is the case in request-response transmission. Otherwise, sensors can be active with a periodic, OnOff or continuous behavior. Considering continuous transmission as a periodic one with a tiny period (ε), IoT events are summed up to periodic and OnOff mode. Concerning the periodic transmission, when the period T elapses, the content is no longer valid. However, with the OnOff transmission, the sensor behavior can not be predicted and the updates of values can be performed at any time. For this reason, the prediction process is used with this mode. We differentiate in Algorithm. 2 the two modes of transmission in the calculation of T_{fresh}. In the case of a periodic sensor, T_{fresh} is the remaining time of the period since the last update (Line 8). With an OnOff sensor, T_{fresh} will be estimated using forecasting tools.

Algorithm 2 Calculation of T_{fresh}

1: **Input:** Received request
2: **Output:**T_{fresh}
3: **Data:**
4: Sensors will receive requests from different consumers to retrieve the sensor's value
5: **Begin**
6: **for** each received request **do**
7: **if** $Data.flow = "Periodic"$ **then**
8: $T_{fresh} = (update\_time + T) - current\_time$
9: **else**
10: Estimate T_{fresh}

5.2.2 Prediction model

For the prediction process, we use time series. With this latter, data is analyzed in order to extract a meaningful statistic model used to predict future values based on the previously-observed ones. There are several forms to model a time series. The exponential smoothing, the machine learning and the Autoregressive Moving Average (ARMA) models are the most widely-used approaches to time series forecasting [Hyndman 2014]. ARMA which is also called the Box and Jenkins approach can be regarded as a special case of the most general and most powerfull algorithm of the Kalman filter algorithm [Caines 1972]. Both are applied only to the processes which satisfy the linear

models with a finite number of parameters. The Kalman filter algorithm carries additional strength, differently from Box and Jenkins approach. One of the main advantages is handling of missing data. In our case, ARMA will be sufficient since all the events will be recorded. The machine learning technique works better if we are dealing with huge amount of data and enough training data is available, while ARMA is better for smaller datasets. The exponential smoothing is suitable for forecasting data which does not display any clear trending behavior or any seasonality. However, the ARMA time series is designed for stationary time series. These latter do not depend on the time at which the series is observed. To understand this point, we depict in Figure. 5.1 a sample of the IoT traffic over the time. In this study, we used real IoT data extracted from ADREAM [LAAS-CNRS 2013] building in LAAS-CNRS laboratory which is a smart building. The building hosts our smart apartment equipped with different sensors (temperature, humidity, luminescence, presence, etc.) as well as actuators such as electric plugs attached to different elements: lamps, fans, humidifier, etc. Figure. 5.1 plots a sensors data flow extracted from ADREAM between 8 am and 11:30 am. As we can show, the figure portrays a time series without trend and we can also note the low variability of the data around their average value. Consequently, we can assume that the time series is stationary.

As a consequence, we choose to use the ARMA model [Makridakis 1997] as a tool for forecasting.

5.2.3 Autoregressive Moving Average model

ARMA model was introduced by Box and Jenkinsis in [Box 1970]. This model is a very large family of stationary processes which has two advantages: on the one hand, these processes are excellent and precise forecasting tools. The forecasting error is proved to be less than or equal to 5% [Brockwell 1991]. On the other hand, methods are perfectly developed to estimate their parameters. The prediction operation in ARMA is based not only on the past events but also on some unexpected recent events. In fact, it consists in eliminating obvious trends as periodicity and growth and then focusing on residue. This latter is modeled and the forecast relied on this model. This model needs a maximum of 30 past values to predict a new one. The size of this set depends on the values themselves [Brockwell 1991]. A process $ARMA(p,q)$ with the notation $\{X_i\}_n$, is presented in equation Eq. 5.1. The different parameters of this equation will be explained later.

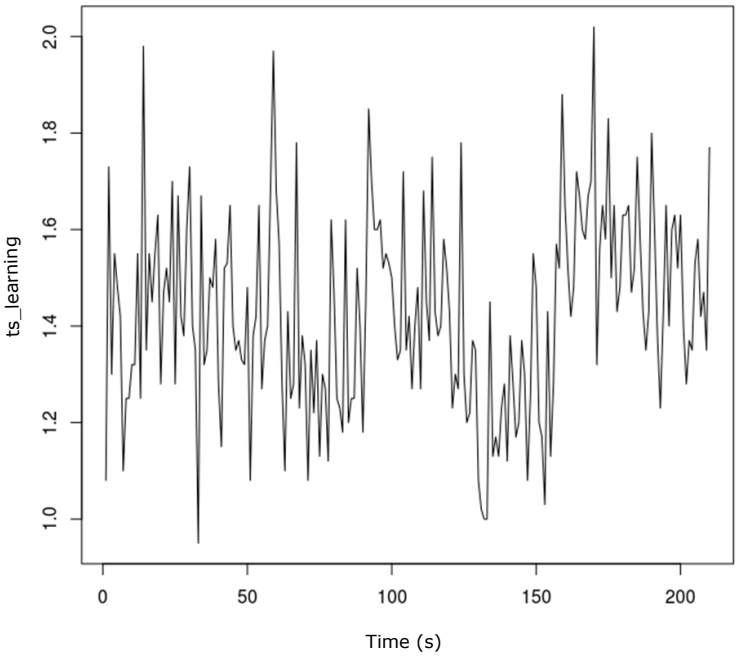


Figure 5.1: The stationarity of the time series

$$\begin{aligned} X_n + \phi_1 X_{n-1} + \phi_2 X_{n-2} + \cdots + \phi_p X_{n-p} \\ = \varepsilon_n + \theta_1 \varepsilon_{n-1} + \theta_2 \varepsilon_{n-2} + \cdots + \theta_q \varepsilon_{n-q} \end{aligned} \tag{5.1}$$

Forecasting T_{fresh} amounts to calculate X_{n+1}. To this end, it is necessary to start by collecting the data $\{X_i\}_n$, then, we calculate ARMA parameters $(p, q, \phi_i, \theta_i, \varepsilon_i)$. The learning phase is indispensable in the forecasting process. It is a primordial step to collect a sufficient set of data and to analyze the followed process in terms of stationarity, trends and seasonality.

5.2.3.1 Data collection

We remind that we want to estimate how long the last recorded value by a sensor will still valid. As a consequence, $\{X_i\}_n$ will be the different time intervals between two consecutive events as portrayed in Figure. 5.2.

We present the data collection in Algorithm 3. We use the queue as a container for $\{X_i\}_n$. In this way, the new values will overwrite the old ones. The size of the queue is

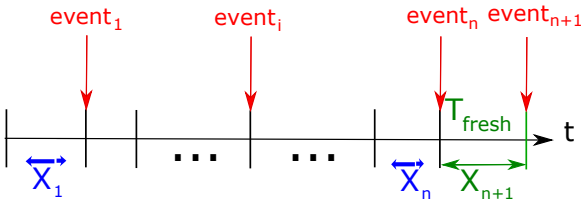


Figure 5.2: Time series model

Algorithm 3 Data collection

Input: A set of sensors
Output:$\{X_i\}_n$
Data:
Each sensor has its *past_events* container
past_events: a queue in a FIFO context which contains $\{X_i\}_n$
Begin
for each sensor **do**
  $time\_interval = 0$
  $last\_event = 0$
  for each event **do**
    if $last\_event \neq 0$ **then**
      $time\_interval = current\_time - last\_event$
      $past\_events.push\_back(time\_interval)$
    $last\_event = current\_time$

set to 30 as we have mentioned previously. Each gateway maintains the queues of the sensors connected to it. For each occurred event in a sensor, we calculate *time_interval* which is the delay between the new event (*current_time*) and the last one (*last_event*) (line 12). Then *time_interval* is pushed into the back of the queue *past_events* (line 13) and we update the *last_event* time with the new event time (line 14).

5.2.3.2 Calculation of ARMA parameters

We remind that ARMA parameters are $p, q, \phi_i, \theta_i, \varepsilon_i$. ARMA process is the sum of the AR process and the MA process. X_n is an autoregressive process of order p, AR_p. It is therefore determined by the variance ε_n of the white noise as well as its canonical polynomial Eq. 5.2, where $\phi_1, \ldots, \phi_p$ are the parameters of the model.

$$X_n = \varepsilon_n - \phi_1 X_{n-1} - \phi_2 X_{n-2} - \cdots - \phi_p X_{n-p} \tag{5.2}$$

The autoregressive model specifies that the future value estimated (X_n) depends linearly on its own previous values $X_1, \ldots, X_p$ and on a stochastic term ε_n which presents the imperfectly predictable term. ε_n is the white noise which measures the variance σ^2 given in equation Eq. 5.3.

$$\varepsilon_n = \sigma^2 = 1/n \sum_{i=1}^{n} X_i^2 - \bar{X}^2 \tag{5.3}$$

The vector $\phi = (\phi_1, \ldots, \phi_p)$ is the solution of the Yule-Walker equation presented in equation Eq. 5.4 where $C_{X,p}^{(n)} = (C_X^n(1), \ldots, C_X^n(p))$ are the empirical covariances (Eq. 5.5) and R_p is the Toeplitz matrix of the covariances to the order p.

$$\begin{aligned} R_p^{(n)} \phi^{(n)} &= C_{X,p}^{(n)} \\ \Leftrightarrow \phi^{(n)} &= C_{X,p}^{(n)} (R_p^{(n)})^{-1} \end{aligned} \tag{5.4}$$

$$C_X^n(k) = 1/n \sum_{j=1}^{n-k} X_{j+k} X_j \tag{5.5}$$

$R_p^{(n)} =$

$$\begin{pmatrix} 1 & C_X^n(1) & C_X^n(2) & \cdots & C_X^n(p-1) \\ C_X^n(1) & 1 & C_X^n(1) & \cdots & C_X^n(p-2) \\ C_X^n(2) & C_X^n(1) & 1 & \cdots & C_X^n(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_X^n(p-1) & C_X^n(p-2) & C_X^n(p-3) & \cdots & 1 \end{pmatrix}$$

We solve the AR_p equation in Algorithm 4. As we can see from line 11 to line 21, p is set according to the calculated ϕ_p. It is a question of choosing the best value of p for the given distribution $\{X_i\}_n$ that will be sufficient for a precise estimation. A corollary used in AR provides a tool for finding the right value of p when we want to model a series with an AR process. We can, in fact, calculate one by one the empirical partial correlations ϕ_p. If we compare the quantile to the 0.975 order of the Gaussian ($Z_{.975}$), divided by $\sqrt{n}$, we can see from which value of p, the value of ϕ_p remains smaller in module (line 15). We can then admit that $p-1$ is the best value (line 17) since p pushes the process in the rejection region.

Algorithm 4 Calculation of the AR_p process parameters

1: **Input:** $\{X_i\}_n$
2: **Output:**X_{n+1}
3: **Data:**
4: Sensors with full *past_event* container
5: Sensors with an OnOff transmission mode
6: **Begin**
7: **for** each sensor **do**
8: Calculate ε_n (Eq. 5.3)
9: **for** each received request **do**
10: p=1
11: **while** $p \leq n$ **do**
12: Calculate $C_{X,p}^{(n)}$ (Eq. 5.5)
13: Calculate $(R_p^{(n)})^{-1}$
14: Calculate ϕ_p (Eq. 5.4)
15: **if** $\mid \phi_p \mid \geq Z_{.975}/\sqrt{n}$ **then**
16: **if** $p \geq 1$ **then**
17: p - -
18: break
19: **else**
20: p++
21: Calculate X_{n+1} (Eq. 5.2)

The $X_{n+1}(T_{fresh})$ obtained with the AR process is already sufficient especially with

perfectly stationary series. However, for better precision, the Moving Average process is used to readjust T_{fresh}. We call the MA process of order q (also called MA_q), the process defined by the equation Eq. 5.6 where $\theta_1, \ldots, \theta_q$ are the parameters of the model.

$$X_n = \theta_1 \varepsilon_{n-1} + \theta_2 \varepsilon_{n-2} + \cdots + \theta_q \varepsilon_{n-q} \tag{5.6}$$

The MA process is combined to the AR one in order to adjust results by taking into account the errors of previous predictions. In fact, ε_i is the error between the past predicted values and the real ones ($\varepsilon_i =\mid X_i^{real} - X_i^{forecast} \mid$). To solve the MA process, we calculate θ_i with the *innovation algorithm*. This latter represents what X_{n+1} brings of something really new to what has been observed until moment n. It is for this reason that we speak about innovation. θ_i is given in equation Eq. 5.7.

$$\theta_i = Cov(X_n, \varepsilon_{t-n})/\sigma^2 \tag{5.7}$$

As the same way with the AR process, we calculate the MA process according to the equation Eq. 5.6, using ε_i and θ_i. It is worth noticing that p and q can have different values. After the calculation of the MA process, we can finally deduce the adjusted value of X_{n+1}. The calculated T_{fresh} is then appended to the requested data and sent back to the consumer. As a consequence, all cached contents will contain the freshness delay information. In this thesis, we have implemented all the algorithms and the ARMA model calculations presented above.

We remind that the added metadata T_{fresh} in all the contents in the network, will be used in two contributions in order to support the data freshness requirement in IoT environments.

5.3 Event-based freshness mechanism

To address the cache coherence problem related to different in-network caching strategies, we conceive a freshness mechanism which we call *event-based* freshness. The role of this coherence mechanism is to check the freshness of a requested data found in a cache node before to send it back to the consumer.

5.3.1 Event-based freshness algorithm

The event-based freshness mechanism is an expiration-based freshness with a variable expiration time. It depends on producer behaviors, the time of storing a content in a cache called cache-time, the period in the case of periodic mode, and past events in the

case of OnOff mode. The pseudo-code for forwarding the request and checking the data freshness is given in Algorithm 5.

Algorithm 5 *Event-based freshness mechanism*

1: **Input:** Request from a consumer
2: **Output:** Fresh Data
3: **Data:**
4: $Data$ to retrieve
5: $Prod =$ the *producer* node
6: $Cons =$ the *consumer* node
7: $G = (V, E)$ a graph that represent the network
8: $Path = (v_1, v_2, ...v_n) \in V * V * ... * V$ s.a:
9: v_i adjacent to v_{i+1} for $1 \leq i \leq n$ and
10: $v_1 \longleftarrow Cons$
11: $v_n \longleftarrow Prod$
12: **Begin**
13: $i = 1$
14: $Found = False$
15: **while** NOT $Found$ AND $i < n$ **do**
16: **if** $Data \in v_i$ **then**
17: **if** $current\_time - cache\_time < T_{fresh}$ **then**
18: $Found = True$
19: Get($Data$)
20: **else**
21: $i = i + 1$
22: Delete($Data$)
23: **else**
24: $i = i + 1$
25: **if** NOT $Found$ **then**
26: Get($Data$) (Data is retrieved from the Prod)

As we can see in the algorithm, the check process is performed when a request reaches a cache that contains the requested $Data$ (Line 16). To verify the content validity, we compare the lifetime of the content in the cache with its lifetime in the producer to check if the cached content has been modified in its source. To calculate the lifetime of a cached content, we use the *cache_time* value stored in the cache node and the T_{fresh} parameter appended to the data. The difference between the current time *current_time* and the *cache_time* measures how long a content was cached. If this latter is less than the freshness delay T_{fresh} then the content is returned to the consumer (Line 17-19). If not, the $Data$ is considered non-fresh and it is deleted from the cache and the request is forwarded to the next hop (Line 20-22). If none of the cache nodes belonging to the

request path maintains a copy of *Data*, the request is satisfied by the producer without a validity check process (Line 26).

5.3.2 Example with event-based freshness mechanism

We describe in Figure. 5.3 a scenario under the Named Data IoT Networks using event-based freshness mechanism. We remind that the event-based mechanism introduces new parameters which are *cache_time* and T_{fresh}. The *cache_time* parameter is stored in the cache structure in the gateway when a new item is cached, however, T_{fresh} is sent as metadata with the content and then stored in the CS structure as shown in Figure. 5.3. In the presented scenario, we consider two producers; a temperature sensor and a presence sensor. This scenario is executed in two phases. The first one is the calculation of T_{fresh} which is performed when a request reaches the producer. The second phase is the validity check which is performed when a copy of the requested content is found in a cache node. At 10:30 am, $Consumer1$ sends a request to retrieve the content named $/Home/room1/Tmp$. The data is satisfied by the producer. This latter is a periodic sensor with a period $T = 1h$ and its last update has occurred at 10 am recording 23° of temperature. As a consequence, the next update will be at 11 am. Since it is 10:30 am, the data will remain valid for $30min$ which is the value of T_{fresh}. When the consumers ask for a content generated by an OnOff sensor, T_{fresh} is calculated using ARMA model. According to Figure. 5.3, the content $/Home/room2/pre$ is retrieved at 10:55 am with $T_{fresh} = 2h$ and its value is 1.

For the second phase, we suppose that $Consumer2$ is interested in content $/Home/room1/Tmp$, so it sends an *Interest* packet towards $n3$. When the *Interest* packet reaches node $n3$, a cache lookup is performed. Then, the content is found in the CS. The use of the event-based freshness consists in verifying cached object before returning it. It is assumed to be 11:30 am. The requested content is cached at 10:30 am with a freshness delay $T_{fresh} = 30min$. That means, after 11 am, the temperature value 23° is considered invalid. Consequently, this entry is removed from the CS, and the request is forwarded to the next hop to look for a valid data. If $Consumer2$ is interested in content $/Home/room2/pre$, the cached copy in $n3$ will remain valid since 12:55 am. We supposed that it is 11:30 am, then we admit that the data is fresh since it is supposed that the next event has not yet occurred. The data is then returned to the consumer.

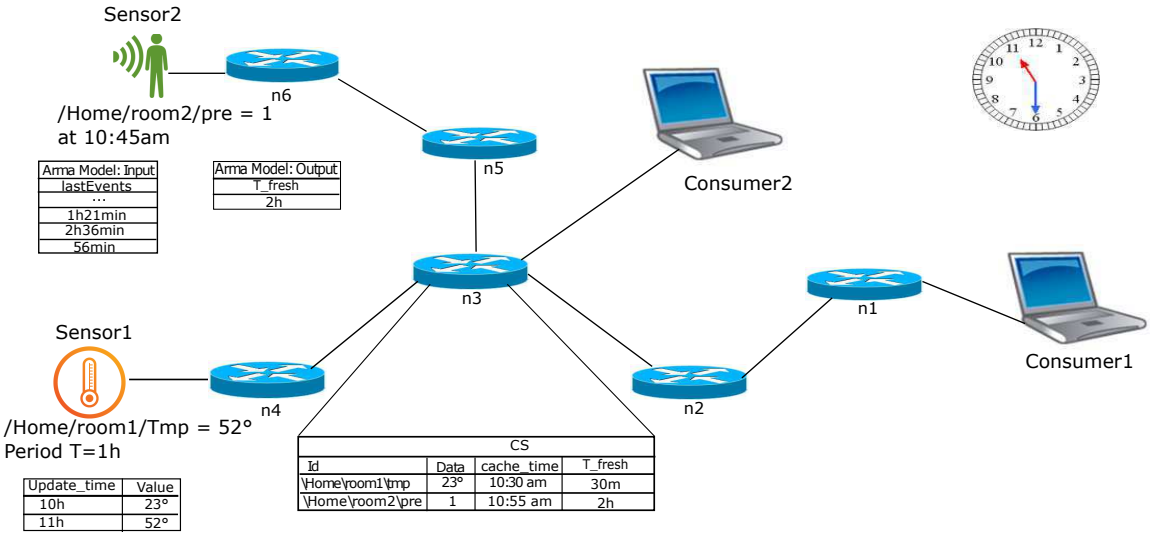


Figure 5.3: An example with the NDN architecture using the event-based freshness mechanism

5.4 Least Fresh First cache replacement policy

Since the cache size is limited, once the cache is full, some stored content must be deleted to allow caching new items. To this end, cache replacement policies are used to select the data object to be evicted. The cache efficiency is strictly related to the adopted policy. In fact, if cached contents are evicted without satisfying any request during their lifetime in a cache or if the cache provides out of date contents, this cache is inefficient. However, in-network caching is basically supported to improve data dissemination efficiency. In order to enhance the cache efficiency, we aim to avoid the eviction of the items that can satisfy future requests and choose to delete already invalid items. To this end, we propose in this thesis a cache replacement policy named the *Least Fresh First* policy. LFF algorithm selects the least fresh data item as the candidate to be evicted from the cache.

5.4.1 LFF algorithm

The approach, presented in Algorithm 6, is invoked at each time when there is a new *Data* to cache. As a first step, a check process is mandatory to see if the content already exists in the cache (line 11). If it is the case, just an update of the *version*, the *cache_time* and T_{fresh} is occurred (line 15-18). If not, it is necessary to verify if the cache is full or not (line 20). If there is a place to add the content, this latter is

stacked above the *Cache* (line 21). In the other case, one data item c_i has to be ejected. Therefore, a look over the *Cache* is performed in order to select the c_i with the least $T_{fresh} + cache\_time$ and evict it (line 23-25).

Algorithm 6 Least Fresh First

```
1: Input: A new data item to cache
2: Output: A data item to evict from the cache
3: Data:
4: Data to cache
5: Cache = (c_1, c_2, ...c_n) s.a: n ≤ cache_size
6: c_i the position of the data item to be evicted s.a: 1 ≤ i ≤ n
7: Begin
8: i = 1
9: Found = False
10: while NOT Found AND i ≤ n do
11:    if c_i.name = Data.name then
12:       Found = True
13:    else
14:       i = i + 1
15: if Found then
16:    c_i.version = Data.version
17:    c_i.cache_time = current_time
18:    c_i.T_fresh = Data.T_fresh
19: else
20:    if n ≠ cache_size then
21:       c_{n+1} = DATA
22:    else
23:       for each C_i ∈ C_{1..n} do
24:          c_evict = min(c_i.T_fresh + c_i.cache_time)
25:          Evict(c_evict)
```

5.4.2 Example with LFF policy

We detail in Figure. 5.4 the operation of the LFF cache replacement policy. We suppose that the cache size is equal to 5. $Consumer1$ is interested in content $/Home/room2\ /pre$. Since there is no entry in cache nodes that can satisfy this request, the data is retrieved from the producer. When the response reaches node $n3$, according to our adopted consumer-cache caching strategy, a copy of the content will be stored in the cache. However, the cache is full, so one item must be evicted to allow the caching of the new one. The candidate to be ejected is the one with the least value of $T_{fresh} + cache\_time$ which measures the time from which the content is considered invalid. In our scenario,

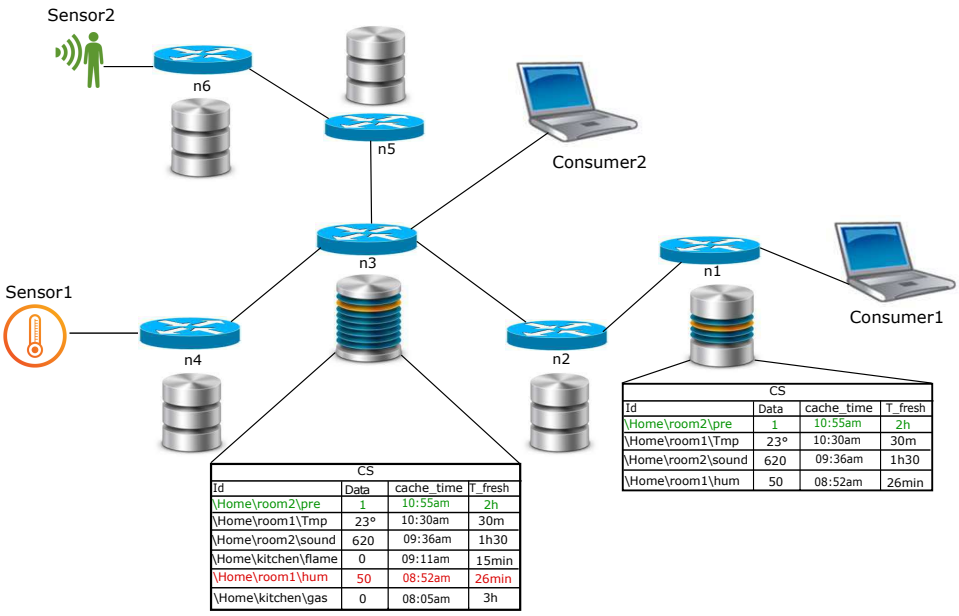


Figure 5.4: An example with the NDN architecture using LFF policy

the content $/Home/room1/hum$ is deleted (red line) and the new item is added to the head on the stack (green line). The CS structure maintained by node $n1$ is not full, then the new item is directly added above the stack. We remark that the cache in $n3$ filled up faster than the cache in $n1$. This is because $n3$ stored the requested content by both $Consumer1$ and $Consumer2$, while $n1$ caches the content of requests from only $Consumer1$.

5.5 Conclusion

In this chapter, we addressed the freshness requirement and we proposed two novel schemes. Our two proposals are based on the prediction model, so we started this chapter by detailing our forecasting tool. Then, we proposed our event-based freshness mechanism which targets to check the data freshness before the retrieval. Finally, we presented our LFF cache replacement policy which is responsible for selecting the least fresh item to delete in a full cache. In the next section, we will address the mobility issue.

Chapter 6

Adaptive Forwarding for Efficient Mobility Support in NDN-based IoT networks

Contents

6.1 Introduction

In the NDN architecture, data objects are requested without any location information and requests are routed to the producer hop by hop according to the forwarding information previously stored in the FIB structure at each node. The producer mobility can cause packets loss due to the impossibility to forward the request to the new location of the content. Therefore, mobile devices are no longer reachable during and after a handoff and FIBs need to be updated.

In this chapter, we focus on data availability requirements threatened by the high IoT network dynamics. We propose a novel and efficient forwarding algorithm in order to support producer mobility. We start this chapter by identifying among different mobility approaches presented in section 3.6.2, the most suitable one for NDN-based IoT networks. We justify that the name-based routing approach can efficiently resolve the producer mobility issue. Therefore, we give an overview about routing and forwarding processes in the NDN architecture. Finally, we introduce our adaptive forwarding algorithm for efficient mobility support.

6.2 Which approach to handle producer mobility in NDN-based IoT networks?

To the best of our knowledge, the support of producer mobility in NDN, in an IoT context, has not been investigated as yet. However, in the web context, as we could see in section 3.6.2, it has started to gain momentum within the research community. The issue is more serious for IoT scenarios because IoT devices are frequently mobile and applications require data to keep continuity. Apart from this specificity, the problem remains the same. Studies referenced in section 3.6.2 could successfully handle producer mobility in NDN. However, in an IoT environment, some of them still remain inefficient in term of the number of exchanged packets in the network, packet delivery path and handover latency. Usually, all approaches require extra exchanged packets to support producer mobility. As a consequence, the objective is to minimize this cost under various system constraints as in IoT networks. In this section, we discuss approaches presented in section 3.6.2, in order to identify which one is adequate to IoT systems.

We remind that NDN is a name-based routing architecture. It uses hierarchical names which can be aggregated to enable the execution of LPM operations. Every node may not have all the information about the content location, however, it limits wrong forwarding decisions in the network. Content names can follow logical or geographical aggregation and in both cases, they do not reflect topological information. Geographical aggregation is usually used for contents stored in the permanent location. For instance, the name $/Home/floor1/room2/tmp$ is given to a temperature sensor located at the room number 2 on the first floor. On the other hand, content with logically aggregated names may be stored in a mobile node. We take the example of an electrocardiogram sensor used to measure the heartbeat of a person with the name $/Bob/health/heart/beat$.

One of the NDN pillars is the unique, persistent and location-independent naming scheme; the consumers are no longer interested in where the content is located. The

locator/identifier separation approach is based on renaming content after the producer mobility and the addition of extra information concerning the location. On the other hand, the location resolution approach is based on the name resolution routing method which also translates names to IP addresses. These two approaches require extra information about the topology included in content objects either in their names or in their payload. Moreover, since this information is topology-dependent, the content object will change when a mobility occurs. As a consequence, handling mobility with the location resolution approach or the locator/identifier separation approach will contradict the NDN nature and would undo its basic strength especially the in-network caching which requires names to be permanent, location-independent and meaningful by themselves. However, the main idea was to deploy IoT under NDN in order to improve the IoT data dissemination efficiency benefiting from NDN features. We then admit that these approaches are not suitable for NDN-based IoT networks.

Concerning the triangular approach, it suffers from a very long packet delivery path, that said, a high response latency. In addition, it requires the update of the intermediate nodes between the old and the new location which increase the processing cost in terms of power and resources consumption.

The routing-based approach is the least investigated approach in the literature to handle mobility. Existing studies use to discover the topology state either at the time of request, or response or both of them.

We summarize in Table. 6.1 the main differences between the four approaches in terms of the routing protocol, the packet delivery cost and the routing path length. We conclude that the name-based routing is the most suitable approach in NDN-based IoT networks. In fact, it respects, as the triangular approach, the name-based routing aspect of the NDN architecture and does not use the location information to forward requests. However, the triangular approach uses a very long routing path. By against, under the routing-based approach, requests are satisfied with an optimal routing path length, as with the location resolution approach. However, this latter undergoes a high packet delivery cost since it requires many exchanged packets to handle the producer mobility. Name-based routing approach can also reduce the number of exchanged packets in the network because it involves *Interest* and *Data* packet in routing handling.

As a conclusion, we assume that a dynamic routing protocol can be an efficient solution that can indirectly handle producer mobility with the respect to the NDN architecture. In addition, it performs a less handover latency and an optimal packet delivery path since packets are routed to nearest copies. However, it also needs a deep study in order to reduce the complexity cost. Therefore, we begin by studying in the

Table 6.1: Mobility producer management in NDN

	Location Resolution	Triangular	Identifier/Location separation	Routing-based
Interest routing	NRS	Named-based	hybrid	Name-based
Packet delivery cost	High	Medium	Medium	Medium
Routing path length	Optimal	Very long	Normal	Optimal

next section the routing and forwarding in NDN.

6.3 Routing and Forwarding planes in NDN

Since the emergence of NDN as a new Internet architecture, routing has experienced a resurgence within the research activities. These studies result in a variety of interesting solutions to deal with routing challenges.

Expect a few restrictions, routing scheme in NDN and IP has the same semantic. In fact, NDN nodes handle a FIB which serves as the forwarding table rather than IP routers and FIBs are indexed by content names instead of IP addresses. Furthermore, during the forwarding process, FIBs propose multiple interfaces for a given entry since requests can be satisfied also by temporary cached copies, however, IP routers just propose a unique interface which is supposed to be the best interface.

The routing task in both NDN or IP is divided into two planes, a control plane and a data plane. The former principal role is to populate forwarding tables by disseminating topology information. In addition, it detects and recovers failures due to network changes. By against, the data plane has a passive role and does not expect any calculations. In fact, this plane is just responsible for redirecting receiving requests to the right interface according to the information stored in the forwarding tables [Yi 2014].

In current IP networks, the control plane is maintained by the routing protocol and the data plane is ensured by the passive forwarding process. Due to the peculiarities of the NDN architecture, such as the support for multipath communications and the presence of a strategy layer allowing nodes to make forwarding decisions on their owns, new questions are raised by research community that prompt the rethink of routing planes role in NDN. Authors in [Yi 2013] have discussed this issue. They affirm that in NDN, the forwarding plane role may not be limited to a passive forwarding but it can stand up with the central role of control plane since it has an active role in making forwarding decisions. As a consequence, new challenges appeared about the interaction of the routing protocol and the intelligent and adaptive forwarding process in managing the control plane in NDN. To this end, the routing protocol is only responsible for disseminating

initial topology and policy information as well as long-term computes the routing table to guide the forwarding process. Concerning the forwarding plane, in addition to its basic role of forwarding requests, it is also tuned for detecting failures and fast recovering them by handling short-term churns in the network [Yi 2012]. As a conclusion, the routing protocol assumes henceforth a bootstrapping role for the forwarding process. It gives a reasonable forwarding information as a starting point for the forwarding plane. This latter can then, in turn, improve the forwarding choices by effectively exploring different interfaces. Authors have proved that exempting routing protocols from failure detection and recovering significantly improves their scalability and stability.

To recapitulate, we present in Figure. 6.1 both planes in the high-level architecture for NDN IoT systems presented before. As we can show, the control and data plane are depicted in the *Core NDN Protocol*. The naming and forwarding process encompass cross-plane functionalities. However, caching and routing strictly belong respectively to the data plane and the control plane.

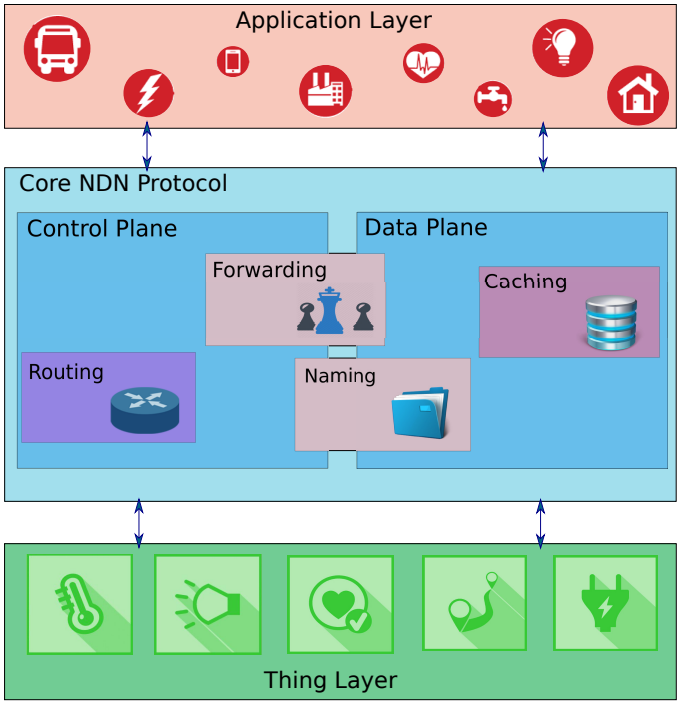


Figure 6.1: The NDN Layer

At this stage, we have presented the different role of routing and forwarding planes in NDN. We report is the following the limits of existing studies for both planes.

6.3.1 Existing studies on routing plane

A few works have explicitly considered the routing protocol targeting to design a protocol to populate FIBs in order to route requests to permanent copies [Wang 2012a, Hoque 2013].

Authors in [Wang 2012a] introduced OSPFN. It is the first study that addresses a routing algorithm designed for NDN and it is based on the well-known Open Shortest Path First (OSPF) protocol. In this protocol, nodes forward Opaque Link State Advertisements (OLSAs) messages with a standard header to build their knowledge of the network topology. Based on OLSAs, the best next hop for each content name can be calculated and added to the FIBs. The same authors have then proposed the Named-data Link State Routing Protocol (NLRS) in [Hoque 2013] in order to improve OSPFN. In fact, this latter still relies on IP while NLRS is entirely based on NDN. The NLRS protocol uses *Interest* and *Data* packets for routing. Furthermore, unlike the current-IP based protocols, routers do not constantly push the routing updates, by against, routers interested on these updates collects topology information with OLSAs based on a receiver-driven model of NDN. Despite the considerable efforts provided by these authors to propose a routing protocol for NDN with the aim to disseminate permanent content locators, they could not avoid the exchange of explicit signaling message to keep location information updates. This can inflate the overhead especially with very dynamic networks as IoT.

Some others studies are focused on how to efficiently forward request towards temporary copies [Wang 2012c, Chiocchetti 2013a, Lee 2011, Liu 2012, Tortelli 2014]. That means, these studies have addressed both planes. Mechanisms presented in [Wang 2012c, Lee 2011, Liu 2012] are based on IP routing as a starting point to populate FIBs. By against, [Tortelli 2014, Chiocchetti 2013a] adopts a pure name-based routing. Some works have proposed to represent the FIB structure with the Bloom filters. Studies in [Wang 2012c, Lee 2011, Liu 2012] use the simple Bloom filter, instead [Tortelli 2014] uses Stable Bloom filter which is also a variety of filters that provides the concept of stability after a certain number of operations (insertion/suppression).

Tortelli. M et al. addressed in [Tortelli 2014] the issue identified with NLRS and they proposed a COntent-driven Bloom filter based Routing Algorithm (COBRA), an intra-domain routing protocol based on a distributed and content-driven approach. COBRA adopts the concept of leaving *Data* traces in NDN nodes to populate FIBs. Authors propose to replace the classical structure of a FIB in each node by SBFs for each interface of the node. As we have already mentioned, *Data* packets in NDN follow the request path to forward back the requested contents to the consumers. When a node receives a

Data packet of a content hierarchically named /A/B/C/D, this information is inserted in the SBF of the incoming interface. The *Data* information is recorded in SBFs as footprints of the whole name as well as its prefixes by progressively eliminating the last component of the name. That means the footprint of /A/B/C/D, A/B/C, /A/B and /A are added. As a consequence, these entries can be used to satisfy future requests sharing the same name or a part of the name using the LPM. It is worth to note that at the starting point all SBFs are empty and consequently no match is found for the first requests. Therefore, in this case, *Interest* packets are forwarded by flooding until founding a match or reaching the producer.

Results have proved that the routing plane in COBRA succeeded at forwarding requests to the right destination. However, this routing algorithm presents some shortcomings. In fact, it is worth to note that the flooding mechanism may result in many response paths, as a consequence, it will be pertinent to consider all paths in order to select the best one, for instance in term of the number of hops. Then, only optimal paths should be stored in SBFs. On the other hand, COBRA substitutes FIB structure by bloom filters for each interface. With bloom filters, we can be certain that an entry does not exist in a list but when we assume that an entry belongs to that list there is a probability of false positive. SBFs also require extra operation of hashing different names and their prefixes for each interface which also undo the human-readability feature in FIBs. FIBs entry may be useful in the analysis of traffic. Furthermore, adding all name prefixes in SBFs can give wrong forwarding decision. In fact, an interface that can satisfy the content /A/B/C/D does not necessarily satisfy the content /A/B/E/F, however, /A/B will be found as an LPM and the request will follow the wrong interface without exploring other interfaces.

6.3.2 Existing studies on forwarding plane

The studies in [Wang 2012c, Chiocchetti 2013a, Lee 2011, Liu 2012, Tortelli 2014] are based on collected information about the existence of different copies in the network and a cost assigned to each incoming interface. For instance, the interfaces cost can be based on retrieval delay [Chiocchetti 2013a] or on the presence of an LPM [Tortelli 2014]. The specific adopted information is gathered using different mechanisms.

Works that have studied efficient forwarding, have proposed a similar idea for link recovery. The main rationale behind these studies is to delete from the FIB the entry related to a content in case of failed transmission in order to explore again different interfaces. For instance, with COBRA, in the case when the request is not satisfied, after a period of time, the consumer will retransmit the same *Interest*, so the node will

detect a retransmission. When a node receives a retransmitted *Interest*, it forwards the received *Interest* towards both the wrong path (in order to delete the entire name along the previous path) and another interface with the LPM. The common drawback in these studies is that proposed link recovery algorithms intervene after the failure to cover it rather than trying to anticipate it to avoid as much as possible the packet loss. In addition, they reuse partially the flooding process to found a new copy. This makes them suffering from a high signaling overhead, due to the high refresh rates of the caches in dynamic networks.

6.4 AFIRM: Adaptive Forwarding based lInk Recovery for efficient Mobility support

In this thesis, we address both routing and forwarding modules and we introduce AFRIM our adaptive routing and forwarding strategies to support the producer mobility. We separately detail, in this section, the FIBs construction phase and the link recovery phase after the producer mobility.

6.4.1 FIBs construction

Facing the limits of existing studies, we propose in this section a routing algorithm to populate FIBs based on the same concept of leaving traces used in COBRA, nevertheless, the implementation details and the complexity are very different. In our proposal, we keep the classical FIB structure for each node and we store entries with their hierarchical human-readable names. We fix a network setup phase in which only flooding is used to forward requests in order to explore all the network. However, in this phase, the tracing process is started and all the data traffic is stored in FIBs. That is, even if there is a match in the FIBs, the requests are flooded during this phase. When a node receives a data packet, it stores in its FIB the entire name and its prefixes as with COBRA. With every entry, it also inserts the incoming and outgoing interface according to the PIT, as well as, the number of hops that the data has crossed from the producer/cache to this node. Since a node can receive the same content from different paths due to the flooding, the number of hops is considered to keep in the FIB the entry coming from the shortest path. In the case when a name prefix is common with all the interfaces, this entry is deleted. For instance, a node that receives the content /A/B/C/D and /A/B/E/F, /A/B and /A/ will be deleted because they are not specific to a unique interface.

We present in Figure. 6.2 the flow chart of the FIBs construction phase. When a node receives a data packet, it first checks if the name already exists in its FIB. We start by checking the entire name then for each iteration the last component is eliminated. If the name does not exist, it is added in the FIB. In the other case, we check if the existing entry is coming from a different interface. In this case, the entry is updated if needed with the least number of hops. If the data comes from a different interface other than that of the existing entry, we differentiate the case if it is about the first iteration. If so the entire name will be inserted with the correspondent interface and the number of hops. If not, this means that the forwarding decision can not be taken since both interfaces have the same probability of satisfying contents with this prefix. In this case, the entry is deleted to allow network exploration. This case is usually reached with the last component because the majority of names in a subnetwork have the same first prefix. Afterward, the last component is eliminated and we start again with the resulting name until it becomes *NULL*. Once the name is *NULL*, the number of hops carried by the content is incremented and the data packet is forwarded to the next hop according to the PIT.

After the setup phase, FIBs are populated with information about requests forwarding. Requests can then be forwarded to the optimal destination based on the number of hops. However, contents availability is not static. In fact, link failure or contents mobility, as well as cache replacement, can lead to packet loss. Therefore, FIBs should be updated to redirect requests through right paths. We consider, in the next section, the producer mobility.

6.4.2 Link recovery

We remind that, in our study, we aim to support the producer mobility. In our proposal, we propose to restore the path in case of content mobility to make it reachable by new *Interests*. We detail in the following our design scheme. AFIRM targets to avoid transmission failure caused by producer mobility. That's why we propose to detect within the optimal time the producer mobility and update the old and the new path. To be aware of the producer mobility, we use keep-alive movement detection method. It is worth to note that not all the sensors allow this method, some sensors just support sending the captured values and do not allow packets reception. With this method, gateways periodically send a ping message to check if the sensor is always attached to it. With the absence of an answer, the gateway affirms that the sensor has changed his point of attachment. Therefore, the forwarding information which leads to this node must be deleted. To this end, a *Recovery* packet is sent by the gateway with the name of

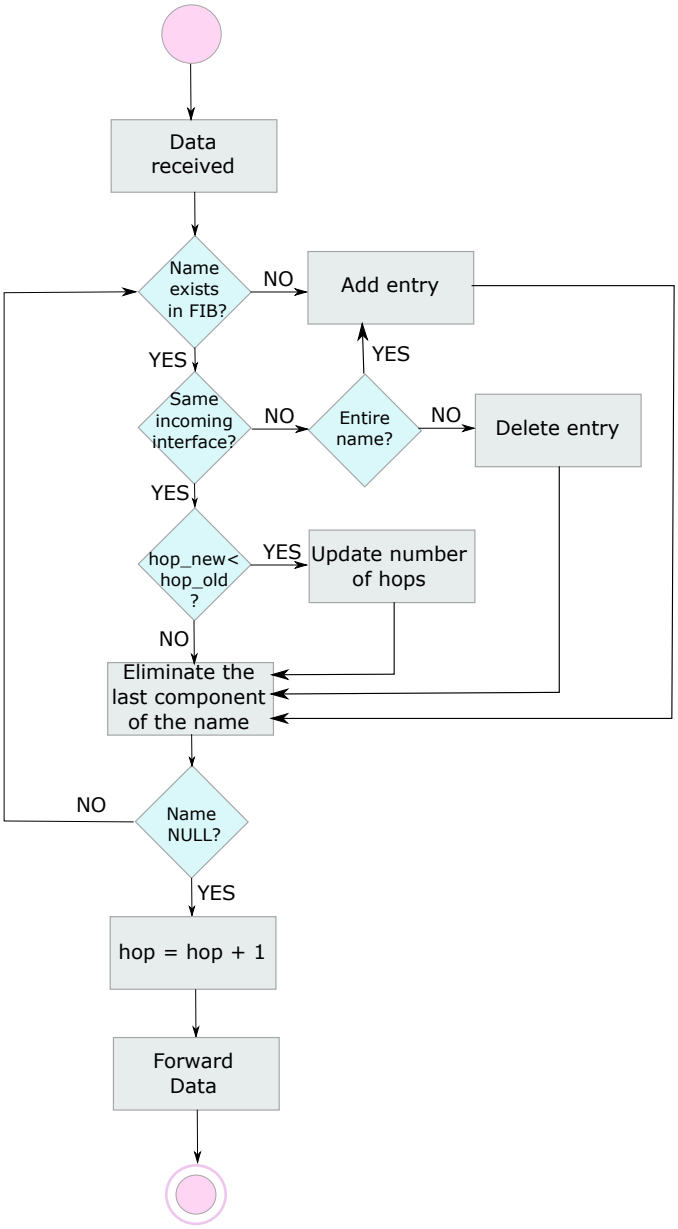


Figure 6.2: AFIRM algorithm: FIBs construction phase

the content(s) handled by this sensor. The *Recovery* packet will follow the corresponding outgoing interface stored in the FIB with this entry and all entries with the entire name will be deleted from the path nodes. On the other hand, the sensor will be attached to a new gateway. This latter will, in turn, send a *Recovery* packet with a positive flag this time to add the new forwarding information based on the LPM and the outgoing

interfaces.

We present in Figure. 6.3 the link recovery algorithm after producer mobility. The figure portrays three flow charts which detail different operation from the point of view of the old location, the new location and other nodes in the topology. We present, in the left diagram, the executed operations when the gateway in the old location detects the disconnection of the sensor. First, a FIB check is performed to be certain that the content has been requested before. If not, nothing is done because in any way there is not a path previously built. In the other case, a *Recovery* packet in created and the flag is set to 0. This *Recovery* packet is then forwarded towards all the outgoing interfaces already stored with the entry. After that this entry will be deleted from the gateway. The diagram in the middle presents the executed operations in the gateway in the new location. When this gateway detects a connexion of a new sensor, it creates a *Recovery* packet with $flag = 1$ and $hop = 0$. A new entry with the entire name is added in the FIB maintained by the gateway. Then, the *Recovery* packet is forwarded to the next hop with the LPM. The last diagram on the right presents the executed operations when a node receives a *Recovery* packet. First, a flag check is performed to know if it is a recovery to delete wrong forwarding information or to create a new forwarding path. When $flag = 0$, this means that the corresponding entry must be deleted. Therefore, we verify if it already exists. The absence of the entry means that the same *Recovery* packet has already passed through this node. In this case, the *Recovery* packet is discarded. If the entry exists, the *Recovery* packet is forwarded towards the interface with the LPM then the entry will be deleted from the node. When $flag = 1$, this means that the corresponding entry must be added. Therefore, the number of hops appended to the packet is incremented and the *Recovery* is redirected to the interface with the LPM then a new entry in added with the entire name, the number of hops and the incoming and outgoing interfaces.

In an IoT context, for better results, we propose to push the last recorded data by the sensor to the gateway to which it is connected and we set the period of the keep-alive movement detection as the updating period T of the sensor or the estimated validity delay T_{fresh}. In this manner, even if the producer moves, requests will be satisfied by the gateway which maintain still valid data and when the period elapses the keep-alive message is sent to check if the sensor is always connected. If not, the recovery process is launched and in the other case, the sensor value is updated in the gateway. The setup phase delay is related to the number of requests. We need to have sufficiently of requests to populate FIBs. We call that the *stabilization_time*.

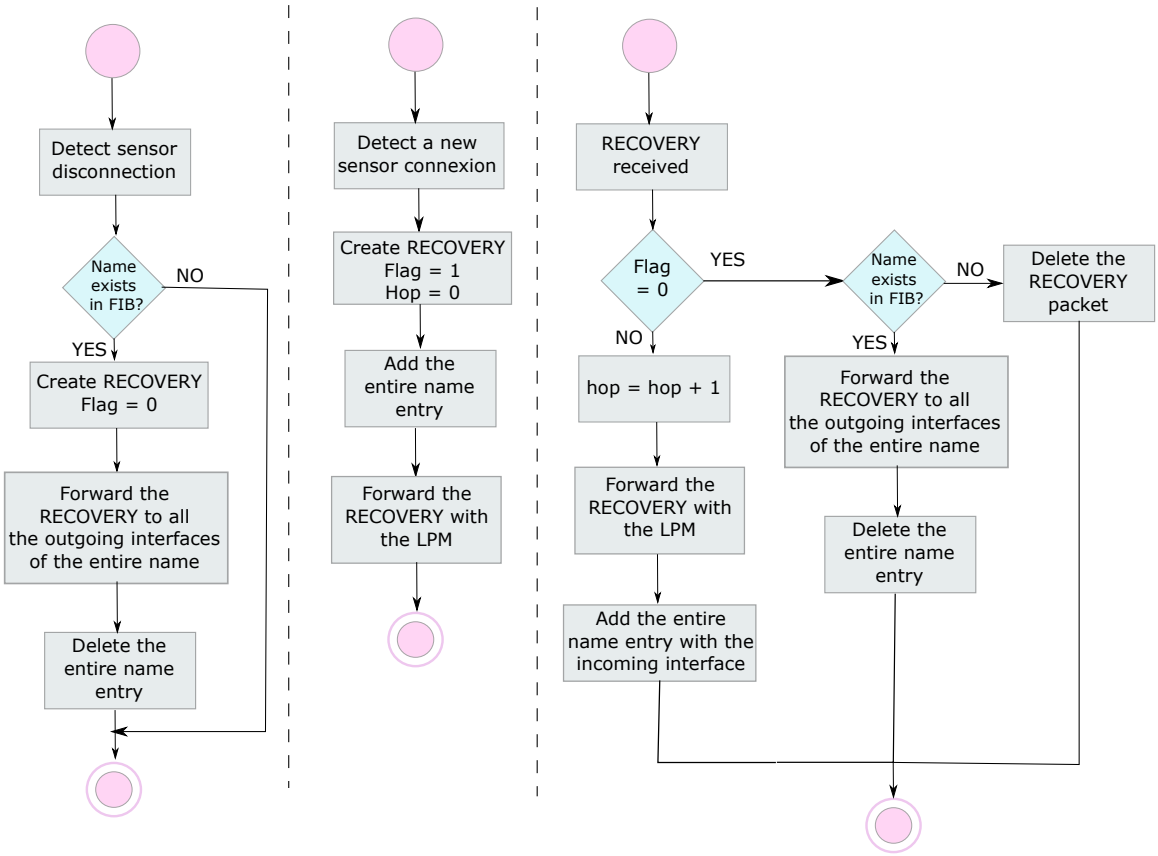


Figure 6.3: AFIRM algorithm: Link recovery after mobility

6.4.3 Example with AFIRM algorithm

For a better understanding, we give in the following section a detailed example of our AFIRM algorithm. We detail, in Figure. 6.4, the FIBs construction phase. Note that red entries are entries to be deleted and green entries are the ones to be added. In this scenario, we suppose that *Consumer1* is interested in content /A/B/C/D. In this phase, *Interest* packets are flooded. When the corresponding *Data* is sent, FIBs in the request path are populated with the name prefixes as shown in nodes *n4* and *n3*. We can remark that, in *n3*, entries that correspond to prefixes /A and /A/B will be deleted. This is due to the request of *Consumer2*. In fact, when the *Data* packet of the content /A/B/E/F reaches *n3*, according to our algorithm, the common prefixes /A and /A/B will be deleted. This to avoid wrong forwarding after the setup phase. Otherwise, if *Consumer2* sends an *Interest* packet to retrieve content /A/G/H/I, when the request

reaches *n3*, it will be redirected to interface 1 since it has the least number of hops for the entry /A (2 rather than 3). We can trivially note that this is a wrong decision.

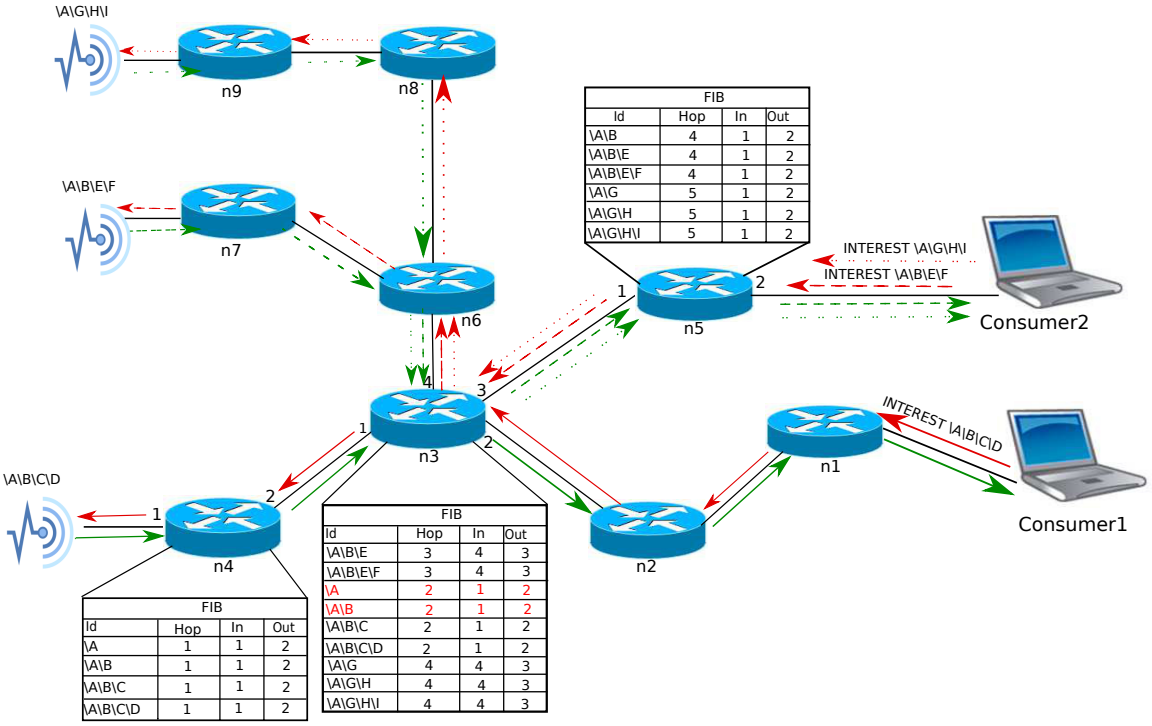


Figure 6.4: AFIRM example: FIBs construction phase

Figure. 6.5 detail the link recovery phase. When the validity delay of a content has elapsed, the direct gateway checks if the sensor is still connected or not. We suppose in this scenario, that sensor A/B/C/D moves from *n4* to *n7*. In this case, *n4* deletes the entry corresponding to this sensor and sends a *Recovery* packet with a null flag to the outgoing interface stored with the entire name. On the other hand, when the sensor connects to *n7*, a new entry is added with the entire name. Then, the *Recovery* is transmitted to the interface 2 which has the LPM and so on until reaching *Consumer1*. In this example, we present the case when the deletion on the wrong paths are done before the adding of the right one. However, it is worth to note that the *Recovery* packet with flag = 1 can reach node *n3* before the *Recovery* packet with flag=0. In this case, the new entry is also added and we will have two entries with the entire name. For this reason, it is essential to identify new entries added by *Recovery* packets with a positive flag when there are two entries with the same name, so that they will not be deleted by the *Recovery* packet with flag = 0. This latter will only discard the oldest

entry and keep the newest one.

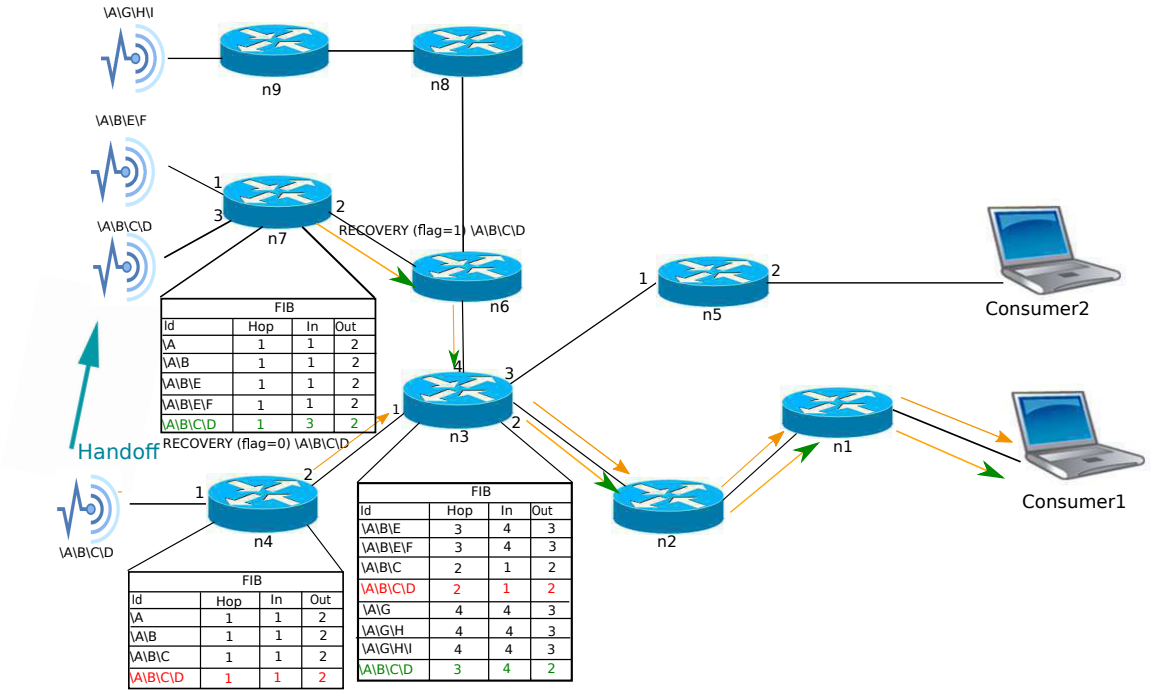


Figure 6.5: AFIRM example: Link recovery after mobility

6.5 Conclusion

In this chapter, we focused on the producer mobility issue in NDN-based IoT networks. We proposed the AFIRM algorithm as a solution to keep data availability in dynamic IoT networks. AFIRM is a content-driven, adaptive forwarding, fully distributed based algorithm for NDN architecture. Our proposal is composed by two phases a FIB's construction phase which is a setup phase used to populate FIB's structures. This first step has a bootstrapping role for the link recovery phase launched after a movement detection.

The next chapter is the last one in this thesis. It is devoted to the evaluation of different contributions. Before analyzing and comparing the results, we will present the simulation setup and the different metrics used to validate our proposals.

Chapter 7

Performance Evaluation

Contents

7.1 Introduction

We evaluate and validate in this chapter our contributions. We compare our proposals against the most dominant and widely used approaches in the literature in terms of in-network caching strategies, validity check mechanisms, cache replacement policies and name-based routing strategies. In the following, all the basic aspects involved in defining the simulated scenarios, such as the network topology, the content catalog and all the fixed parameters are described in detail. Then, we present the metrics on which we are based to evaluate our proposals. Finally, obtained results are thoroughly analyzed and commented.

7.2 Simulation setup

For our simulations, we use the ccnSim simulator [Chiocchetti 2013c]. It is a C++ framework under the OMNeT++ discrete-event simulator that implements the routine

to simulate the NDN architecture. Every node implements the three NDN structures in form of layers. The first one, called "Core layer" is responsible for both the PIT management and the communication with the different layers. The second one, cache layer, represents the CS parts in the NDN structure. It acts according to a caching and replacement strategies. The third element is the strategy layer that takes the decision about *Interest* forwarding. We start the evaluation chapter by this simulation setup section. We will detail the used topology. Then, we present the set of IoT data generated during the simulation as well as the choice of the configuration parameters.

7.2.1 Topology

Concerning the topology, authors in [Pentikousis 2016] affirm that there is not a single topology that can be used to evaluate ICN aspects and this choice depends on the focus of evaluation. In our simulations, we present results with Transit-Stub (TS) topology whose properties imitate closely the IoT topology. The TS graph is a 3-level hierarchical topology presented in [Calvert 1997]. This model is composed of interconnected stub and transit domains and LANs connected to stub nodes. The first level presents the backbone which connects the transit domains. A stub domain carries only the traffic that originates or terminates in the domain. However, transit domains consider all transmissions and their role consist of efficiently interconnecting stub domains. The TS parameters are T, which is the total number of transit domains in the backbone, S is the total number of stub domains per transit node. N_T and N_s are the average numbers of nodes per transit and stub domain respectively. L is the average number of LANs per stub node, and N_L is the average number of hosts per LAN. We set $T = 2$ with $N_T = 10$, $S = 2$ with $N_s = 6$, $L = 1$ and $N_L = 1$. The total number of nodes is $N = TN_T(1 + SN_S) = 260$ nodes and the total number of hosts $N_H = TN_TSN_SLN_L = 240$ hosts. Figure. 7.1 presents the model of our TS topology.

Figure. 7.2 portrays the real topology captured from our simulator. As it is presented, producers and consumers can be connected to 240 hosts. We choose to distribute them in such a way the producer and its consumer do not belong to the same transit domain, however, a host can connect a consumer and a producer at the same time. We generate the topology with GT-ITM[1] (Georgia Tech Internetwork Topology Models) using the parameters fixed above. GT-ITM is a complete set of tools for the conversion of network topologies that support NED language used in OMNET++.

Transmission delays are set by GT-ITM. Values are in the range of $[2; 78]\,ms$. They are set so that transmissions in the third level are faster than the second level and the

[1] http://www.cc.gatech.edu/projects/gtitm/

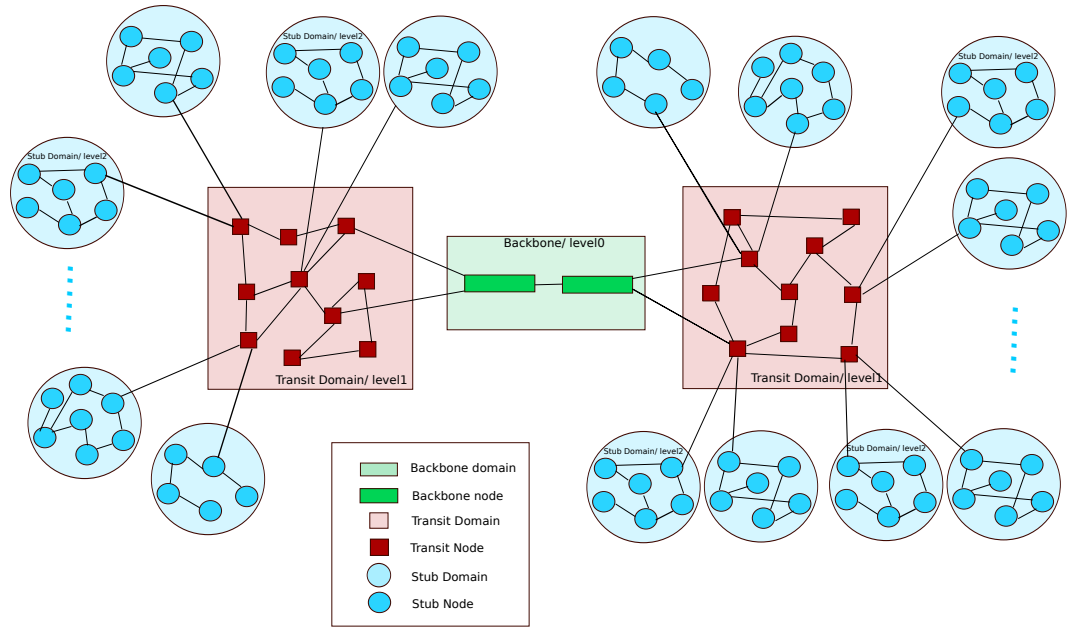


Figure 7.1: Transit-Stub topology model

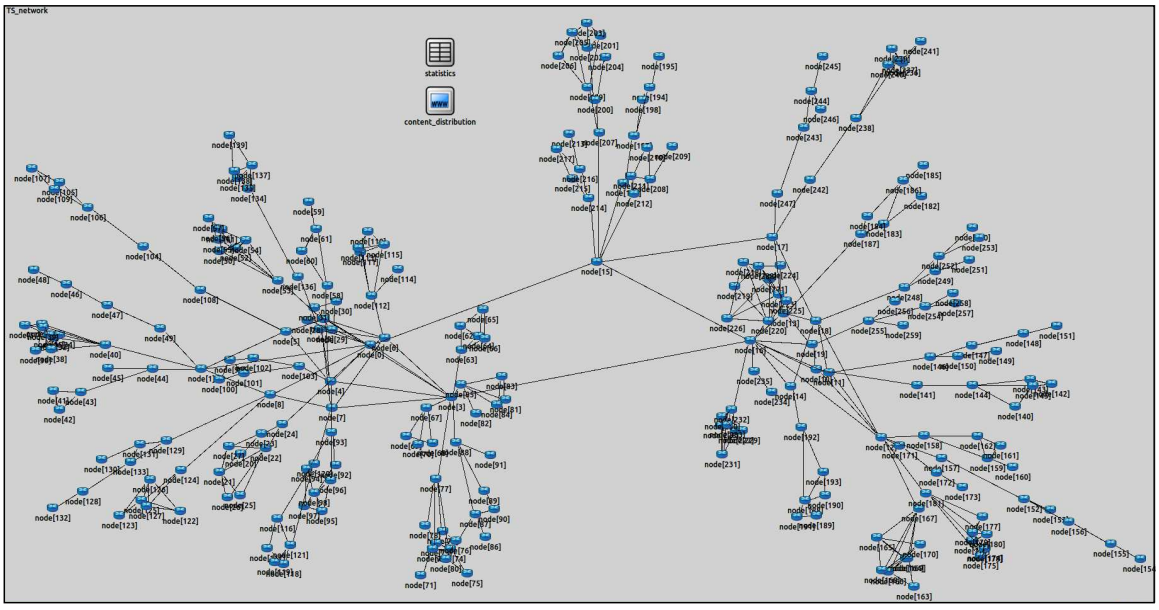


Figure 7.2: Real Transit-Stub topology

same with the second and the first level.

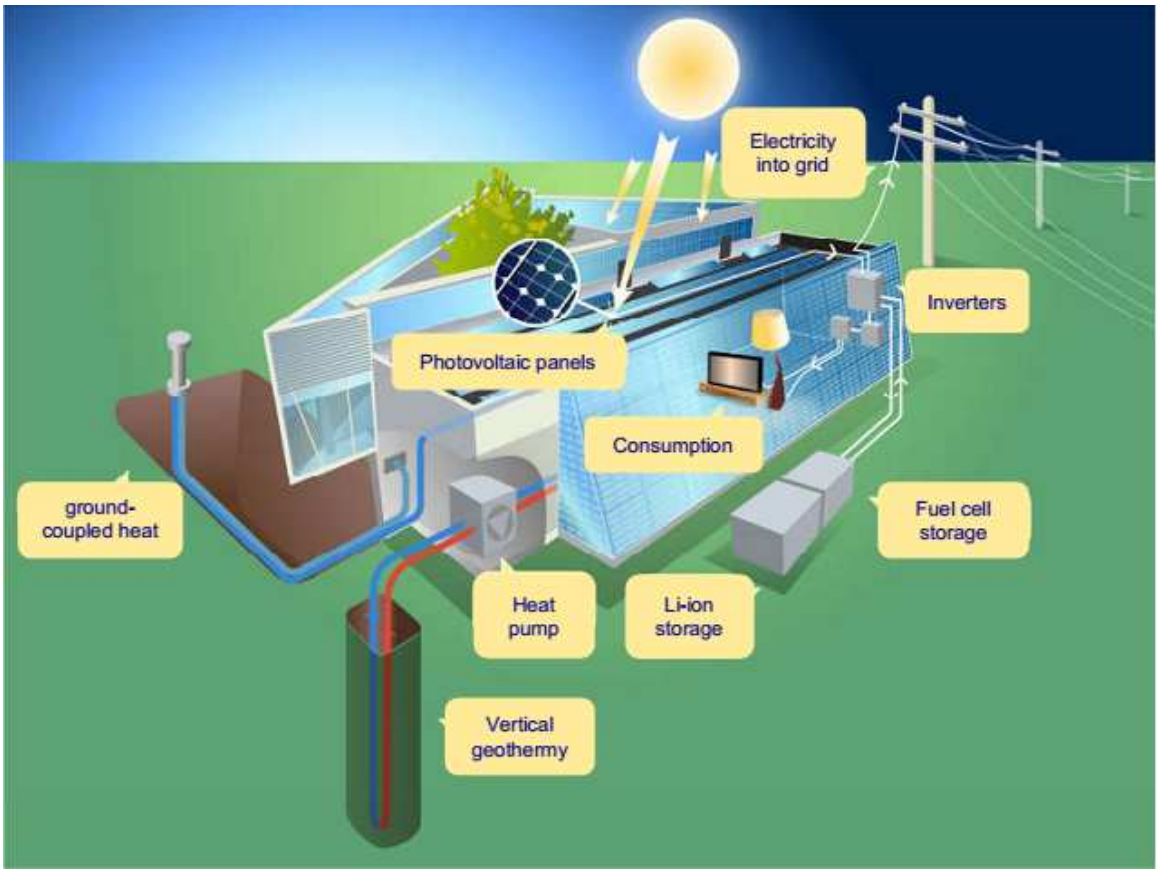


Figure 7.3: ADREAM building

7.2.2 Content catalog

Our simulations were carried out with a real IoT data extracted from the ADREAM [LAAS-CNRS 2013] building in LAAS-CNRS laboratory which is a smart building. ADREAM is a complex cyber-physical environment. As shown in Figure. 7.3, it is composed of many devices including heating, ventilating, and air conditioning (HVAC) devices, heat pump system, photovoltaic cells, weather station, robots, cameras, luminaries, smart meters and many other specific sensors and actuators. The building hosts our smart apartment equipped with different sensors (temperature, humidity, luminescence, presence, etc.) as well as actuators such as electric plugs attached to different elements: lamps, fans, humidifier, etc.

We choose periodic sensors with different periods T varying from $1min$ to $1h$. We used smart meters such as temperature, humidity and luminescence. Concerning the

OnOff sensors, we used devices having different variance, for example, a presence detection sensor injected in a corridor has not the same update frequency as a presence detection sensor injected in a bedroom. We used presence, vibration, fall and smoke detection sensors.

7.2.3 Parameters configuration

We summarize in table 7.1 the system parameters used in our simulations. Analyzing the request popularity distribution in different geographical locations, S. K. Fayazbakhsh et al. deduced, in [Fayazbakhsh 2013], that the web distribution, used by all works on ICN, behaves as a Zipfian distribution. Indeed, the majority of ICN studies use the Zipf distribution which stipulates that some popular contents have a high probability to be requested (e.g., new films, news, today's weather, etc.). However, under IoT, we do not refer to this distribution since it is devised for web-based contents and Internet applications. In fact, in an IoT scenario, contents have close probabilities to be requested. Therefore, in our case, *Interest* packets are uniformly distributed. Studies in [Quevedo 2014b, Hail 2015] which consider IoT environment also used the uniform distribution.

We note that each producer provides a single content as a single chunk with a unique replica in the network. The number of producers is then equal to the number of contents $\mid F \mid$ in the network and the file size F is equal to 1 chunk. In [Rossi 2011], D. Rossi and G. Rossini show that in existing studies the ratio of the cache size C over the catalogue size $F \mid F \mid$, $\frac{C}{F|F|} \in [10^{-5}; 10^{-1}]$. In our simulation, we set $\frac{C}{F|F|} = 10^{-3}$.

Taking into consideration this ratio, we set the file number $\mid F \mid = 4000$ files and the cache size $C = 4$ chunks. The 4000 sensors are connected to 40 Gateways. We consider a variable number of consumers ranging from 20 to 30 and we suppose that all consumers are already connected at the beginning of the simulation. Consumers ask for files following the arrival rate of the Poisson process with $\lambda = 1$.

7.3 Evaluation metrics

We have previously introduced the cache cost metric to address IoT requirements. However, it is primordial to maintain the system performances while improving the cost and the data freshness. In order to validate our contributions, we consider the *hop reduction ratio*, the *server hit reduction ratio* and the *response latency* metrics.

The *hop reduction ratio* α measures the reduction of the number of hops traversed to satisfy a request compared to the number of hops required to retrieve the content from

Table 7.1: System parameters

Parameter	Meaning	Values
C	Cache size	4 chunks
$\mid F \mid$	Producers	4000 sensors
F	File size	1 chunk
$\frac{C}{F\|F\|}$	Cache size and Catalogue size ratio	10^{-3}
$Cons$	Consumers	$[20; 30]$ consumers
λ	Arrival rate	1
R	Replicas	1
$transmission\_delay$	Transmission Delay	$[2; 78]\, ms$
$simulation\_time$	Simulation Time	$200s$
$Events$	Content Catalog	ADREAM DATA

the server. α is analytically represented by the equation Eq. 7.1

$$\alpha = 1 - \frac{\sum_{i=1}^{N} \frac{\sum_{r=1}^{R} \frac{h_{ir}}{H_{ir}}}{R}}{N} \tag{7.1}$$

Where N is the number of consumers, and R is the number of requests created per consumer. α represents the average over N consumers of averages over R requests per consumer of the hop reduction ratio of the request r sent by consumer i; $\frac{h_{ir}}{H_{ir}}$. The h_{ir} parameter is the number of hops from the consumer to the cache that satisfies r, and H_{ir} is the number of hops from the consumer to the producer.

β represents the *server hit reduction ratio*, the second metric that measures the reduction of the rate of access to the server. In other words, the alleviation rate of the server load. The equation Eq. 7.2 calculates this metric, where $serverhit_i$ is the number of requests sent by i and satisfied by the server (producer) and $totalReq_i$ is the total number of requests, satisfied by both the server and the cache, sent by i.

$$\beta = 1 - \frac{\sum_{i=1}^{N} serverhit_i}{\sum_{i=1}^{N} totalReq_i} \tag{7.2}$$

The third metric is the *response latency*. It is the duration between the delivery of a request and its response. In equation Eq. 7.3, we calculate γ, the average of the response latency T_{ir} over N consumers and each one sends R requests.

$$\gamma = \frac{\sum_{i=1}^{N} \frac{\sum_{r=1}^{R} T_{ir}}{R}}{N} \quad (ms) \tag{7.3}$$

7.4 Simulation results

This section details the performance evaluation of our contributions. Our findings are given in two sets; static scenario and dynamic scenario. The first set considers the contributions that are not impacted by the network dynamicity. By against, the dynamic scenario concerns the contribution designed for mobile nodes.

7.4.1 Static scenario

Concerning the static scenario, we evaluate consumer-cache caching strategy, event-based freshness mechanism and LFF cache replacement policy.

7.4.1.1 Consumer-cache caching strategy results

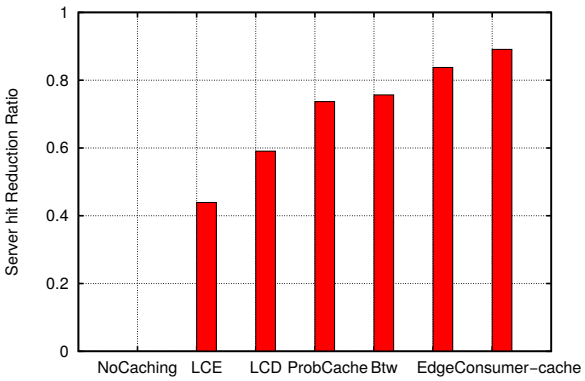
We start with evaluating different caching strategies and compare them to our consumer-cache strategy in term of system performances. CcnSim is distributed with native support of LCE, LCD and ProbCache caching strategies. We have enhanced the simulator and we have implemented Btw, Edge and consumer-caching strategies.

Figure. 7.4 portrays the system performances: the Server hit reduction ratio (Figure. 7.4a) and the hop reduction ratio (Figure. 7.4b) using 25 consumers and Figure. 7.4c portrays the response latency as a function of the number of consumers (varying from 20 to 30). From the latter, we notice that the response latency gets better as the number of consumers increases. Indeed, increasing the number of consumers leads to an increase in the number of requests, which in turn raises the contents availability inside the topology.

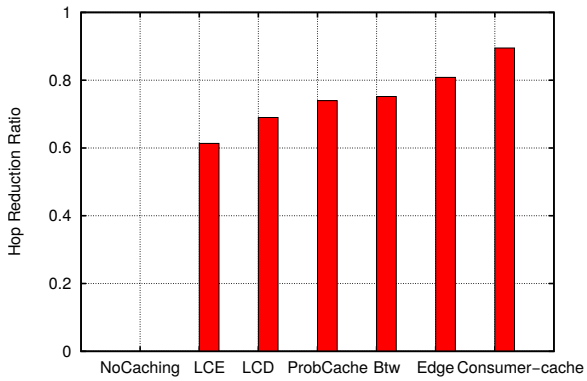
Without a caching strategy the Server hit Reduction Ratio and the Hop Reduction Ratio are equal to zero as all requests are satisfied by the producers.

The LCE strategy stores copies everywhere, which make content available at every node. However, caches fill up quickly and consequently, old contents are rapidly evicted which increases the number of evictions and leads to cache misses. This explains the fact that this strategy performs the worst results in this scenario. To clarify the Server hit reduction ratio results, we plot in Figure. 7.5 the average number of evictions for each caching strategy.

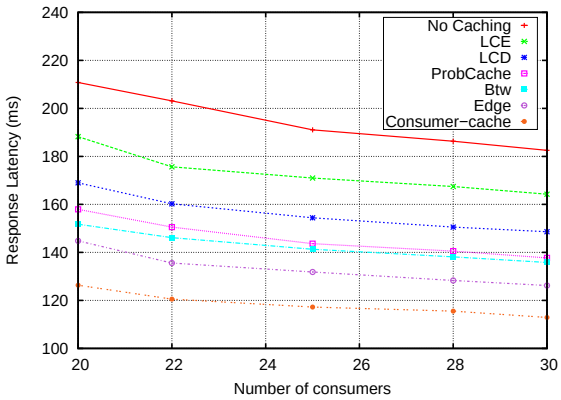
We notice that the increase in the number of evictions diminishes the cache efficiency. The LCE strategy has the highest number of evictions as it was expected. Figure. 7.5



(a) Server hit reduction ratio



(b) Hop reduction ratio



(c) Response latency

Figure 7.4: System performances for in-network caching

shows that the closer the cache nodes are to the producer the higher the number of evictions is. This is because nodes close to the producers belong to many request paths by against nodes close to consumers belongs only to request paths starting from this consumer. Since the LCD strategy selects the cache nodes at one level down from the producer, it has a high number of evictions. With ProbCache, contents can be cached on more than one node within the request path probably near to the consumer. However, with Btw, there is one cache node in each request path and generally in the middle of the path. ProbCache and Btw have a very close number of evictions. This latter is slightly lower under Btw. Finally, the edge-caching and Consumer-cache have almost the same number of evictions.

With LCE caching strategy, The Server Hit Reduction Ratio (Figure. 7.4a) is 0.43. This value means that only 43% of requests are satisfied from cache nodes. The Hop Reduction ratio (Figure. 7.4b) for this strategy is about 0.61. Which means that paths

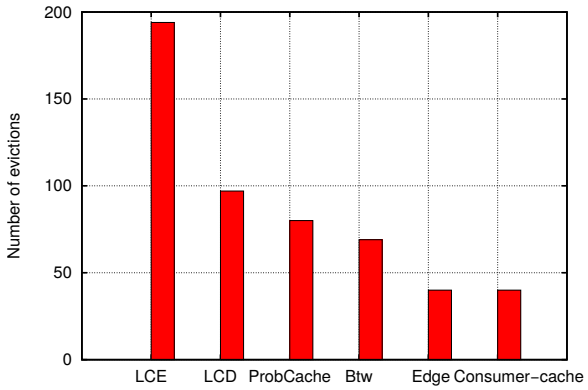


Figure 7.5: Number of evictions

are reduced by 61% in term of the number of hops. Concerning the third metric, the response latency (Figure. 7.4c) is about 182ms to 210ms.

The caching strategy LCD decides to cache contents at the node on one level down from the response source (Producer/cache node). After a certain number of requests, it tends to LCE and all path nodes become caches. For this reason, LCD results, are not so good like with LCE. The Figure. 7.4b shows that LCD records 69% of hop reduction, and requests take about 164ms to 188ms of response latency (Figure. 7.4c). Its server hit reduction is about 0.59.

Concerning ProbCache and Btw, cache nodes are selected in the middle of the request path and probably more close to consumers, in the case of ProbCache. Simulation results of these two strategies are medium comparing to other caching strategies. The hop reduction ratio using ProbCache and Btw is respectively about 0.73 and 0.75. The same for response latency, ProbCache and Btw report respectively about 137ms to 157ms and 135ms to 151ms. Figure. 7.4a shows 0.73 of server hit reduction under ProbCache and 0.75 for Btw.

We remind that the third assumption on which we are based was that the edge nodes are the best placement for cache nodes. Our findings confirm the results presented in [Fayazbakhsh 2013]. In fact, edge-caching reports good results. Under this strategy, we measured 0.83 in server hit reduction, 0.80 of hop reduction ratio, and 126ms to 144ms as response latency.

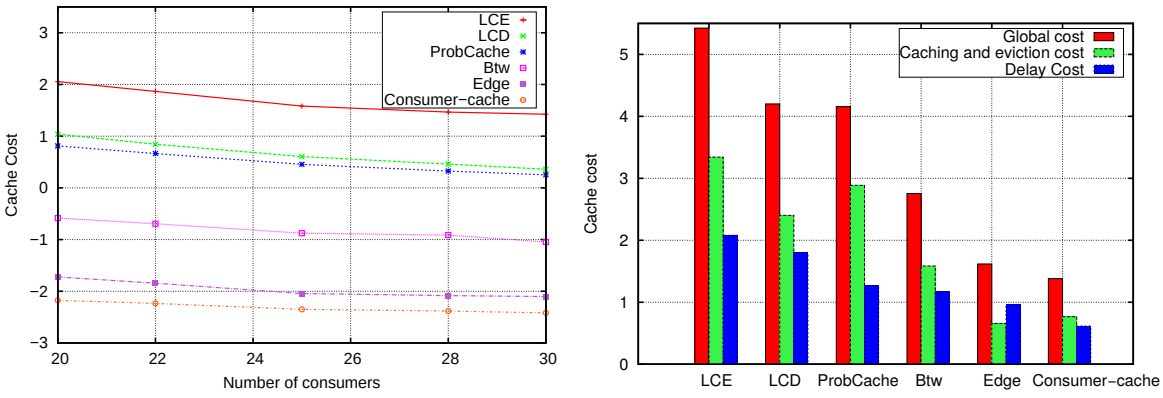
We detail now the results of our consumer-cache strategy. This latter stores copies in nodes attached to consumers which allow these consumers to easily reach requested contents. Consumer-cache has the best simulation results because requests are, in most cases, satisfied by the first hop node. We report for our strategy 0.89 of the server hit

reduction. The hop reduction ratio is about 0.89, this implies that requests only cross 11% of hops on the path towards the producer. Finally, with our strategy, the response latency varies from 112ms to 126ms. The minor difference between consumer-cache and edge-caching strategies stems from the fact that consumer-cache strategy makes contents more close to consumers and requests generated by consumers not located in the topology leaves are satisfied by the producer under edge-caching.

As it was shown previously, in-network caching efficiency strictly depends on the number of cache evictions. In fact, this parameter impacts the system performances since the cache replacement may cause cache misses. However, the impact of the number of evictions on the performance results is not proportional. For instance, edge-caching and consumer-cache have the same number of evictions but consumer-cache outperforms edge-caching. In fact, under edge-caching, *Interest* packets sent by consumers in the middle of paths will be satisfied by the producer since there is no cache node within the request path. In this case, we have a cache miss without any cache eviction. However, consumer-cache makes contents available to consumers, just in one hop.

Furthermore, we may assert that, in high traffic environment like IoT, we should minimize packets transmission within the first (backbone) and the second topology levels. Our proposed consumer-cache strategy, as well as the edge-caching strategy, significantly relieve these two levels as most of the requests are satisfied within the stub domain.

Since caching strategies can be quite costly in term of used resources, we calculate the cache cost introduced in chapter 4 of all caching strategies. The Figure. 7.6 illustrates the cache cost function.



(a) Cache cost

(b) Caching cost and delay cost

Figure 7.6: Global cache cost

Figure. 7.6a plots values of the reduced centered normal distribution Z for the cal-

culated values of the cache cost function X (Eq. 4.4). Negative values are the result of the normalization method. Simulation results are obviously positive.

We illustrate, in Figure. 7.6a, the cache cost with different caching strategies. We notice that the increase of the number of consumers leads to a slight decrease in the cache cost. The same curve was depicted in Figure. 7.4c and this is because the cache cost also includes the system performances in term of delays as shown in equation Eq. 4.3.

Caching strategies have different cache costs. In fact, we expect that LCE has the higher caching and eviction costs seen that it stores copies everywhere and cache nodes are more numerous. In addition, LCE has the worst system performances. From this view, LCE will have the worst global cache cost. In our previous results, LCD strategy doesn't perform good system performances. Furthermore, this strategy has many cache nodes because for each request the content is solved at one level down. As a consequence, LCD, in our scenario, is a costly strategy. Also with the ProbCache strategy, the cache cost is assumed to be high. This is due to the fact that system performances are medium and content can probably be cached in different nodes for each request. Under Btw strategy, the caching and eviction costs are expected to be minimal, because, always it is the same node, with higher betweenness centrality, that keeps a copy of a content. In the other hand, system performances with Btw strategy are medium. This makes its cache cost medium. The edge-caching strategy had good system performance results as consumer-cache, in addition, the number of cache nodes in the topology are constant and not very high. Consequently, the cache cost under this strategy is not very high. Concerning, our caching strategy, consumer-cache performs the best system performances, and moreover, cache nodes are limited as with edge-caching, which makes it the least costly strategy.

As we have discussed, we report, in Figure. 7.6a, from 2 to 1.4 cache cost as the highest cost with LCE. The minimum cost is calculated with our consumer-cache strategy with -2.4 to -2.1. The second least costly strategy, edge-cache, reports -2.1 to -1.7. Then, Btw, has -1 to -0.5 of global cache cost. At the last, we found that ProbCache and LCD have similar results with 1 to 0.25.

For better understand of the trade-off between the caching cost and delay cost, we separately depict in Figure. 7.6b the delay cost and the caching and eviction cost with 25 consumers. The objective is to show which of the different components building up the Global Cost metric dominate.

The Figure. 7.6b shows that the delay cost is less than the caching and eviction cost. Furthermore, we remark that with LCE, the cache cost is very important. In the other side, the edge-caching strategy has the least caching and eviction cost since it has the

least number of cache nodes. Consumer-cache strategy has also a very low caching and eviction cost and its low delay cost makes it the least costly caching strategy. We can also remark that although ProbCache system performance results were better than LCD results, these two strategies have similar cache cost results. With Figure. 7.6b, we can understand that ProbCache has well a low delay cost, however its caching and eviction cost is higher than the LCD one which makes them equal in term of global cost.

In an IoT environment, system performances, as well as cache cost, need to be closely considered. Our proposed consumer-cache performs the best trade-off between content availability and caching cost. As such, consumer-cache strategy stands out as a viable solution in a such IoT environment. In the following section, we evaluate our second contribution, which is the event-based freshness mechanism.

7.4.1.2 Event-based mechanism results

To evaluate the freshness mechanism, we use the following metric presented in the equation Eq. 7.4. *Validity* is the percentage of the valid contents received by N consumers and satisfied by cache nodes against the total number of received content including valid and invalid ones. In the equation Eq. 7.4, we respectively note $valid_i$ and $invalid_i$ as the number of valid and invalid content received by consumer i and satisfied by a cache node.

$$Validity(\%) = \frac{\sum_{i=1}^{N} valid_i * 100}{\sum_{i=1}^{N} valid_i + invalid_i} \tag{7.4}$$

With the aim of maximizing the content validity percentage to meet the IoT data coherence requirement, we propose to integrate our freshness mechanism to several caching strategies. Event-based freshness mechanism tries to predict the times of updates in order to eliminate copies supposed to be invalid. Figure. 7.7 depicts the percentage of fresh content with or without freshness mechanisms using different caching strategies.

Without freshness mechanism, cached copies are never checked before being sent. After a certain amount of time, all copies will be deprecated. As it is shown in Figure. 7.7, the percentage of content validity for all caching strategies, without the use of freshness mechanism, do not exceed 20%. We report, 2% using the LCE, LCD, Btw and ProbCache strategies, 9% with edge-caching and 19% under consumer-cache. We notice that the consumer-cache strategy inherently maintains data validity compared to other caching strategies. This result depends on the number of evictions (Figure. 7.5). In fact, strategies that have a high number of evictions may remove from caches still valid contents. In addition, thanks to the high performance, especially the response latency,

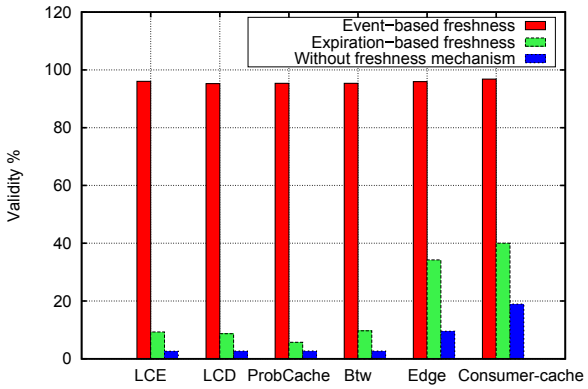


Figure 7.7: Validity % with freshness check mechanisms

with the consumer-cache the number of retrieved content is greater than other strategies during the simulation time.

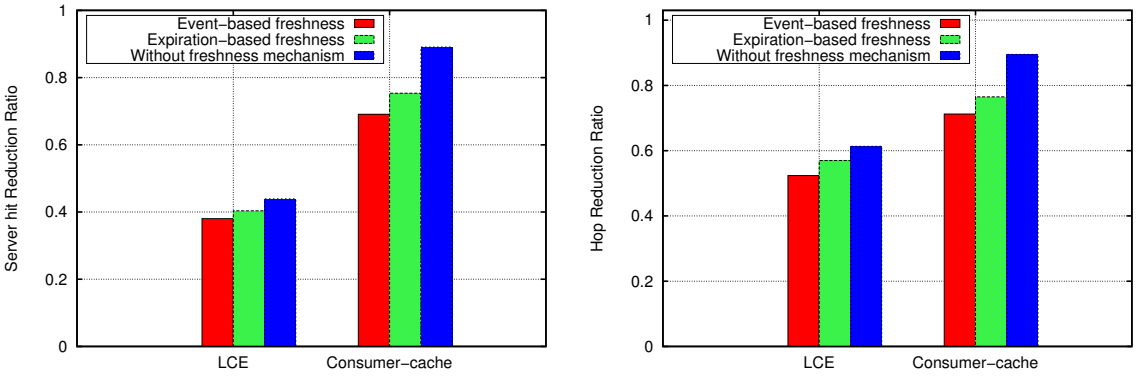
Expiration-based freshness mechanism comes to deal with this problem, but it is difficult to fix the right *expiration_time*. We propose to put the average of events period in order to cover the maximum of content updates. We choose $expiration\_time = 20s$. Under Expiration-based freshness mechanism, the improvements are not very impressive. The graph portrays 9% using LCE, LCD and Btw, 5% with ProbCache, 34% under edge-caching and 40% using consumer-cache.

Our freshness mechanism has proven that it can significantly improve the content validity percentage. Figure. 7.7 shows that this percentage can reach 98% with different strategies. We conclude, from this figure, that event prediction is a good solution to increase the content validity, especially with steady systems.

We infer that even if the expiration-based freshness mechanism combined with our consumer-cache strategy improved the validity percentage, the event-based freshness still performs better results. Without any freshness mechanism, the eviction influences the data freshness due essentially to adopted replacement strategy which may delete still fresh contents near to consumers and keep non-fresh ones in the network.

To validate our proposal, it is essential to look at the impact of the freshness check mechanism on the system performances. We report in Figure. 7.8 the system performances with and without the freshness check mechanisms under LCE and consumer-cache strategies.

We plot in Figure. 7.8 the server hit reduction ratio and the hop reduction ratio. The results report a slight degradation in the system performances when we use a freshness check mechanism. This is because, with freshness mechanism, more nodes are traversed



(a) Server hit reduction ratio

(b) Hop reduction ratio

Figure 7.8: System performances with freshness mechanisms

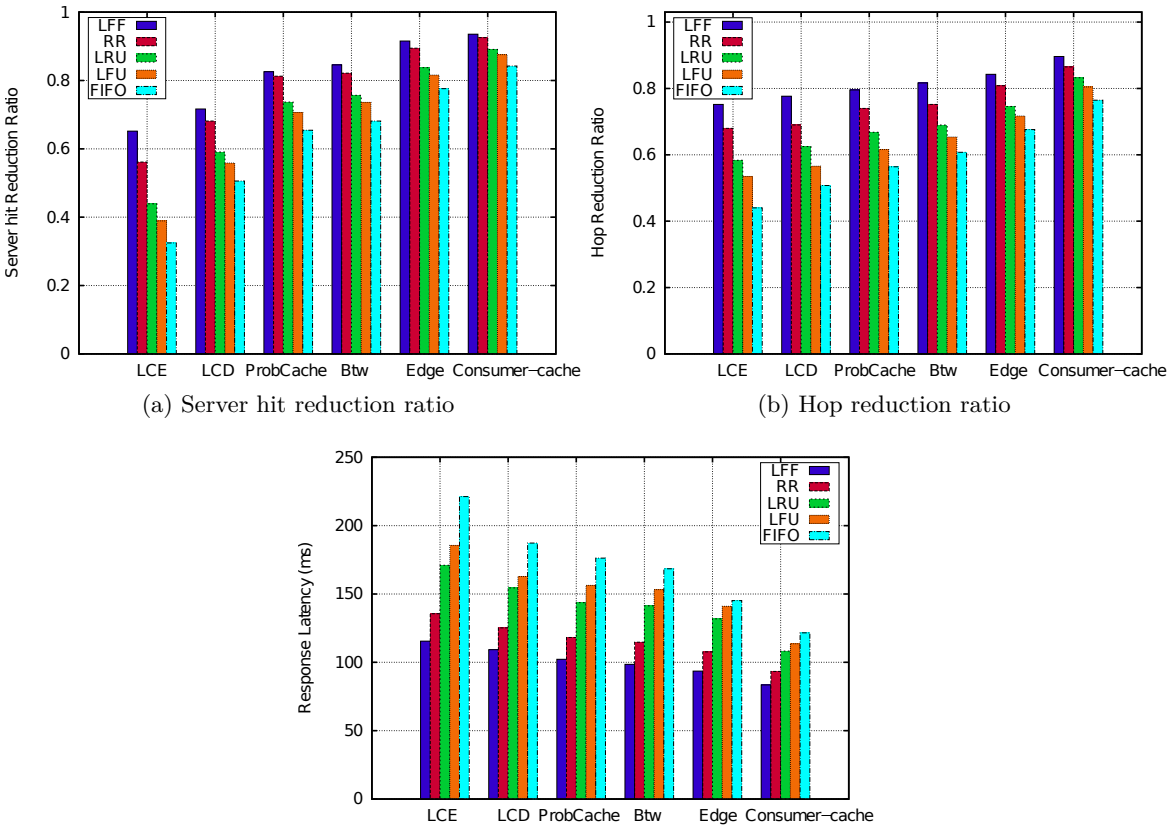
to look for a valid content while without the freshness check the first found content is retrieved. We can also remark that the impact of the freshness check is more important under the consumer-cache. In fact, this latter has fewer cache nodes in the topology. If the first CS does not match the requested data, probably the content will be retrieved from the producer. By against, LCE has many cache nodes in the topology. Furthermore, the expiration-based mechanism performs better results than the event-based mechanism. This comes back to the precision of the prediction using ARMA. In many cases, the expiration-based mechanism admits that the data is valid while it is not.

With the event-based freshness mechanism, we measured in Figure. 7.8a about 0.52 under LCE and 0.71 under consumer-cache of server hit reduction ratio. Concerning the hop reduction ratio, our proposal reports 0.38 using LCE and 0.69 with consumer-cache (Figure. 7.8b).

Facing this degradation in the system performances while trying to satisfy the freshness requirement, we propose the LFF cache replacement policy in order to make the cache more efficient. We will detail in the next subsection our LFF results.

7.4.1.3 Least Fresh First cache replacement policy results

We aim, in this subsection, to evaluate our proposed cache replacement policy LFF. Therefore, we ran experiments with different combinations of caching strategies and cache replacement policies. In our scenario, we suppose that all contents have the same size. In addition, in IoT networks, contents are generally small since they represent sensors' values. So, we will not consider the SIZE cache replacement policy. CcnSim is



(a) Server hit reduction ratio

(b) Hop reduction ratio

(c) Response latency

Figure 7.9: System performances for cache replacement policies

distributed with native support of LRU cache replacement policy. We have enhanced the simulator to support RR, LFU and FIFO cache replacement policies.

Figure. 7.9 plots the system performances, namely, the Server hit Reduction Ratio in Figure. 7.9a, the Hop Reduction Ratio in Figure. 7.9b and the Response Latency in Figure. 7.9c. The first impression one might have by looking at these figures is that different cache replacement policies have the same behavior under different caching strategies.

Based on the values of system performances previously depicted with LCE and Consumer-cache, we can also note that the difference calculated using various cache replacement policies is much larger with LCE than with consumer-cache. In other words, we note that the impact of different cache replacement policies is much more noticeable with LCE than Consumer-cache. This point is explained by referring to Figure. 7.5 which stipulates that when the number of evictions, under a specific caching strategy

is not very important, the choice of the used cache replacement policy does not significantly impact the results. In fact, with the consumer-cache caching strategy and the edge-caching strategy, the number of evictions is very low. Since the replacement process is invoked at the eviction time, the cache replacement policy is not frequently used.

In the following, we evaluate the impact of different cache replacement policies independently of caching strategies. Figure. 7.9 shows that LFF and RR policies outperform LRU, LFU and FIFO policies. It is worth noticing that these findings are strictly dependent on the request distribution. In fact, as we have already mentioned in the simulation scenario, web applications use a Zipfian distribution in which requests privilege popular contents. As a consequence, it is more advantageous to keep in caching nodes the most requested contents. In this sense, LRU and LFU were designed. These policies may perform better results in the web environment. By against, in an IoT environment, requests are uniformly distributed and all sensors have close probabilities of being solicited. In other words, contents are randomly requested. This explains why, in our scenario, the RR policy outperforms LRU and LFU. The FIFO policy aims to keep each content as long as possible in the cache node regardless of the frequency with which each content is requested and the evicted item is not uniformly selected. This policy may be suitable for closed queue-based request distribution. In our scenario, FIFO presents the worst results. Concerning our proposed cache replacement policy LFF, the selection process of the content to be evicted is not related to the incoming requests but to the data freshness. For this reason, our policy does not contradict a uniform request distribution. This explains why RR and LFF findings are very close to each other. The minor difference between these two policies is due to the fact that the RR policy does not follow any logic. It can delete a content which has just been stored or rather the opposite, keep in the cache a content for a long time. The LFF is more coherent vis-a-vis the contents lifetimes.

Figure. 7.9a reports from 0.65 to 0.93 of server hit reduction under LFF and from 0.56 to 0.92 with RR policy. Under LRU policy, from 43% to 89% of requests are satisfied by cache nodes. This figure portrays between 0.38 and 0.87 of server hit reduction using LFU. Finally, with FIFO policy, results are almost from 0.32 to 0.84. Figure. 7.9b gives the same performance results. LFF policy outperforms other replacement policies with 0.75 to 0.89 for hop reduction ratio. RR performs close results, about 0.67 to 0.86. This ratio is about 0.58 to 0.83 and between 0.53 and 0.80 under LRU and LFU respectively. With FIFO policy, requests traverse from 24% to 56% of the path towards the producer. The response latency, depicted in Figure. 7.9c, is the lowest with LFF policy with 83ms to 115ms. It is about 93ms to 135ms under RR policy. With LRU and LFU, it respectively

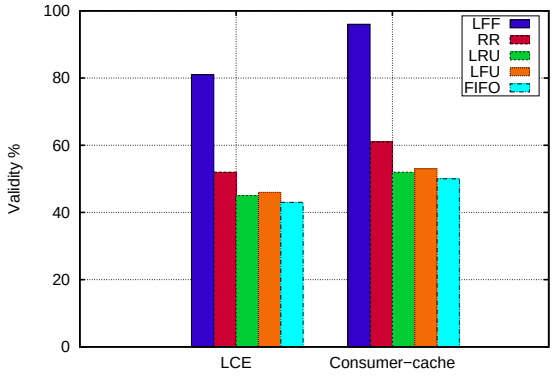


Figure 7.10: Validity % with cache replacement policies

varies from 108ms to 170ms and from 113ms to 185ms. The FIFO policy reports the longest response latency with 121ms to 221ms.

The system performances results have validated our proposed cache replacement policy since it respects ICN goals and in-network caching efficiency. On the other hand, we remind that our policy was designed with the aim of maximizing the content validity percentage to meet the IoT freshness requirement. LFF policy tries to predict the exact delays of updates in order to eliminate copies supposed to be invalid. Figure. 7.10 depicts the percentage of fresh content with different cache replacement policies under LCE and consumer-cache strategies.

As it is shown in Figure. 7.10, LRU, LFU and FIFO have almost the same percentage of content validity, about 52% with consumer-cache and 45% with LCE. In fact, LRU and LFU policies usually keep in the cache for a long time more solicited contents until they become invalid. The same with FIFO, it follows the logic to keep in the cache all the contents for the longest time. The RR policy has slightly better results by comparing it to LRU, LFU and FIFO, about 61% with consumer-cache and 52% with LCE. This policy is random and does not follow any law to manage the lifetime of a content in a cache. Concerning our proposed LFF cache replacement policy, it calculates the required lifetime for which the content is supposed to be valid. Then, at the eviction time, if all the contents are valid, it selects the one who has the least lifetime remaining and if many contents are already invalid, it selects the one who was invalid for the longest time. Our proposal has proven that it can significantly improve the content validity percentage. Figure. 7.10 shows that this percentage can reach 96% with consumer-cache and 81% under LCE. We notice that the consumer-cache strategy better maintains data validity than LCE. This result comes down to the inefficiency of caching with LCE. We infer that

LFF is a good solution as a cache replacement policy for NDN-based IoT environment which targets data freshness.

To recapitulate, we proved that our both proposals namely the event-based mechanism and LFF cache replacement policy can perfectly maintain the data freshness of IoT data. We have reported 98% of the percentage of content validity using the event-based freshness mechanism to check the data validity before the retrieval. The 2% of invalid contents are due to the prediction error of the ARMA model. Under LFF, the validity percentage is about 96%. With this cache replacement policy, invalid contents are evicted one by one, which can keep in the cache invalid data since the cache can maintain more than one invalid content. As a consequence, the 4% of invalid contents are due to the prediction error but also to the remaining invalid contents after the eviction. However, the event-based mechanism slightly degrades the system performances since requests are not satisfied by the first found content but needs to cross more hops to found a valid content. By against the LFF improves the system performances since it makes the cache more efficient. We deduce that the combination of these two proposals may give good results. The event-based algorithm will increase the data freshness to reach 98% and the LFF cache replacement will improve the system performances by providing valid content near to the consumer.

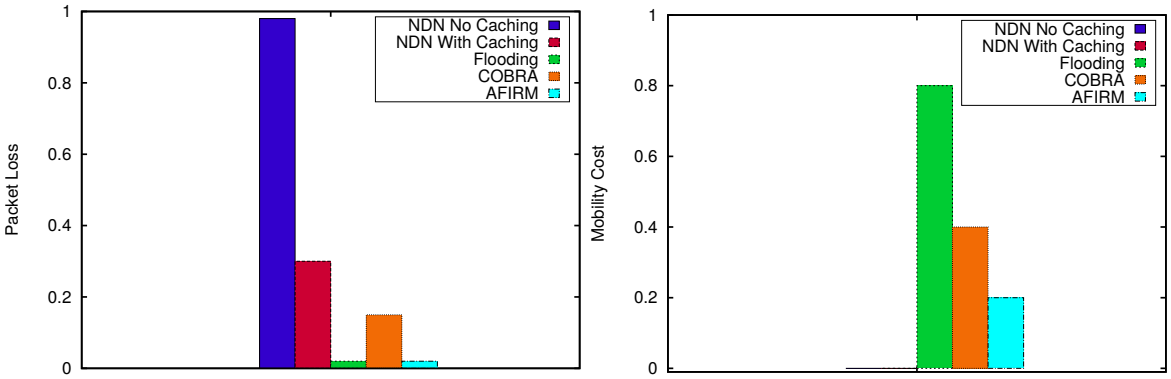
7.4.2 Dynamic scenario

In order to evaluate our proposal to handle the mobility, we need to consider a dynamic scenario that model producer mobility. Therefore, we consider the Random Walk model [Camp 2002]. Based on this model, mobile producers move randomly and freely without restrictions. We set the $stabilization\_time = 20s$ which is a sufficient time in our scenario to populate FIBs with forwarding information.

To assess the performance of AFRIM algorithm, we measure the number of packet loss and the cost of handling producers mobility. The number of packet loss is translated by the number of retransmitted *Interest* packets over the total number of *Interest* packets sent by all the consumers in the topology. Concerning the mobility cost, it represents the extra amount of traffic generated to support the mobility over the total traffic. For instance, with AFIRM, the mobility cost is the ratio of the number of *Recovery* packets over the total number of NDN packets (*Interest*, *Data* and *Recovery*).

Figure. 7.11 portrays the results of our simulations. This figure presents the packet loss ratio (Fig. 7.11a) and the communication cost due to mobility support (Fig. 7.11b).

The results of AFIRM is compared against the flooding solution and the COBRA algorithm. We have also considered the results when the mobility is not supported and



(a) Packet Loss (b) Mobility Cost

Figure 7.11: Mobility support results

we report the findings while considering the two cases with and without the in-network caching. In fact, caching can partially solve the producer mobility issue since requests can be satisfied by copies stored in cache nodes rather than the mobile producer. As we can show in Figure. 7.11a, with the default NDN architecture, the number of packet loss is very high, almost 0.98. This is because, after the handover, contents are no longer reachable and only the first requests before the mobility are satisfied. However, when caching is enabled, cache nodes found in the request path can respond with data previously retrieved. The packet loss is about 0.3. It is worth to note that this result is strictly dependent on the adopted caching strategy as well as the freshness requirement of the consumer. After a certain period of time, cached data can be out of date or can be evicted from the cache, consequently, caches can become useless and results will be similar to those without caching. The mobility cost is, of course, null with these two approaches. The flooding is the basic solution to handle the mobility and it is very efficient. In fact, it represents a naive forwarding strategy which aims to explore the topology for each *Interest* packet. Each node automatically forward received *Interest* to all other interfaces. As consequence, requests are always satisfied. The packet loss with the flooding approach is almost null (0.02). The few retransmitted *Interest* packets are due to other network problems as the link failure. On the other hand, as it is expected, the flooding solution reports a very high communication cost which is about 0.8. Our findings have confirmed that the name-based routing solution COBRA can solve the mobility issue. COBRA can significantly reduce the packet loss. Only 20% of the transmitted *Interest* packet are lost. This loss is due to the fact that COBRA launches the failure recovery after the *Interest* packet loss. The signaling packets used to

update the forwarding information represent 40% of the total exchanged packets. These packets include packets used to delete the wrong information and packets used to found a new path towards the content. Our proposal outperforms other approaches. Under AFIRM, the packet loss ratio is almost null as with the flooding solution. In fact, AFIRM anticipates the failure and recover it before any received packet. In addition, the recovery cost value is about 0.2. AFIRM is less costly than COBRA. In fact, these two solutions have nearly the same cost to delete the wrong forwarding information. However, to add the new one, AFRIM proceeds on upstream with the LPM, by against COBRA add the information on downstream taking into account more than one interface to explore the network. In addition, with AFIRM, by going up towards consumers, intermediate nodes give more precise forwarding decision which avoids the flooding. We conclude that our proposal can significantly support the producer mobility by reducing the packet loss. Furthermore, the signaling overhead is reasonable.

7.5 Conclusion

This chapter covered all the simulations results. We first validated our consumer-cache caching strategy. We proved that this latter improves the system performances in terms of hop reduction ratio, server hit reduction ratio and response latency. In addition, the consumer-cache strategy is a non-costly caching strategy which makes it suitable for IoT ecosystem. Then, we evaluated our event-based freshness mechanism and we proved that it can satisfy IoT data freshness requirement. The same for the LFF cache replacement policy, it succeeds to maintain the percentage of validity and furthermore it was proved that LFF can improve the caching efficiency since it improves the system performances. Finally, we analyzed the performance of our adaptive forwarding algorithm named AFIRM, designed to support the producer mobility. We demonstrated that AFIRM is a good solution for dynamic IoT networks. It almost eliminates the packet loss while significantly reducing the signaling overhead.

Chapter 8

Conclusion and perspectives

The ICN paradigm promises to be the architecture of the Future Internet to handle its evolution towards a vast, heterogeneous and infinite world of equipment, consumers and services. The rationale behind this concept is to address contents by their names rather than their hosts addresses. As the father of ICN, Van Jacobson, said: "*The direct, unified way to solve these problems is to replace where with what*". Thanks to the native support of multicast, in-network caching, name-based routing and easy data access, the research community argues that named data is a better abstraction for today's communication problems than named hosts.

IoT is the Internet infrastructure which most suffers from the massive traffic and the communication complexity. This is due to the increasing number of smart things and the stringent requirements imposed by their applications. Therefore, ICN has attracted the attention of the IoT community inciting the use of the information-based concept in IoT networks. This thesis showcased the potential of ICN as a solution to support IoT systems. Among several proposals of ICN architectures, the so-called NDN architecture has proved to be the most suitable information-centred architecture for IoT environments. NDN defines a receiver-driven, pull-based, robust connection-less communication model, providing an easy and scalable data access, energy efficiency and mobility support.

Despite the numerous benefits reaped from the use of NDN in IoT networks, the requirements set by IoT applications slow down the emergence of the information-centric paradigm in the world of IoT. Indeed, this latter is firstly a heavily resource-constrained environment. On the other hand, it requires a very high data availability and a low response latency. The in-network caching is the most important feature that can deal with the IoT applications expectations. However, the caching strategy should make content always available to the consumers without increasing the amount of data and the redundancy in the network. Furthermore, since IoT data are transient and frequently updated, contents residing in cache nodes will eventually become out of date. Moreover, some critical IoT applications impose a stringent requirement of data freshness. In addition, it is worth to note that IoT devices are usually mobile. As a consequence, producers are no longer reachable after a handover. This specificity requires a dynamic

and adaptive routing approach to be always able to reach mobile producers.

Summary of our contributions

The main objective of this thesis is to improve the data dissemination in IoT networks. We primarily gave an overview of the ICN paradigm and its different building blocks and we detailed the evolution of the IoT and its requirements. Afterward, we presented different ICN architectures in order to prove that the NDN approach is the adequate architecture for the IoT environment. In this thesis, all our contributions are carried out on an NDN-based IoT network.

In the first contribution, we focused on the in-network caching strategy. We proposed the *consumer-cache* caching strategy with the aim to make contents available and promptly accessed by the consumers while reducing the number of caches in the topology. Results proved that our proposal can improve the system performances in term of response latency, hop reduction ratio and server hit reduction ratio. In addition, it results in significantly lower cache cost comparing to existing caching strategies.

Our second and third contributions addressed the freshness requirement. We started by proposing a validity check mechanism before the data retrieval. This mechanism called *event-based freshness* mechanism is based on time series for event prediction model. This latter is used to check if the cached content has been updated in their sources or not. Findings showed that our proposal leads to a notable improvement in content freshness. We reported about 98% of data validity, however, it has slightly degraded the system performances in order to found valid data to satisfy requests. Facing this degradation, we introduced *LFF* cache replacement policy with the aim to make the content stores more efficient. To this end, we decided to avoid evicting still valid content and delete non-fresh ones. The LFF policy is also based on time series analysis as a tool to predict future events. Findings showed that our proposal exhibits superior results in term of system performances and about 96% of freshness percentage of the retrieved contents compared to existing cache replacement policies. The combination of these two contributions maintains both the system performances and the data validity.

The fourth contribution addressed the producer mobility issue. We proposed the forwarding *AFRIM* algorithm to efficiently populate routing tables and adaptively update forwarding information after a producer mobility. The obtained results proved that our proposal can support the mobility feature of IoT environments. Furthermore, it outperformed other solutions as flooding and the COBRA algorithm. Our findings showed that AFIRM can significantly reduce the packet loss due to the mobile sensors with the

respect to the signaling communication cost.

Perspectives

Despite the ICN-based solutions proposed in this work to deal with IoT challenges, there is still room for improvement of our contributions and still challenges to address. We list, in the following, short and long-term perspectives to give potential enhancements of our study and future research directions in ICN-based IoT networks.

The short-term improvements of our work are related to the evaluation methods of our contributions. To evaluate the cost of our proposed caching strategy, we were based on an analytical expression that measures the tradeoff between the caching and eviction cost and the data availability. This evaluation can be improved to assess the real impact on the IoT devices. For instance, the decrease of the charge level of the device battery. In addition, contributions that address the freshness requirements are based on the calculation of the prediction model. The cost of these calculations is not considered in this study. To validate this method it will be interesting to quantify the forecast cost in the gateways directly connected to the sensors. Finally, the dynamic scenario is modeled by the random walk mobility model. A real mobile traffic for a specific application domain as the vehicular transport will give more realistic results.

Concerning the long-term perspectives, we propose some points. First, the caching strategy in ICN can also be context-aware. In fact, considering the name semantic of retrieved data, we can make a decision to cache or not a copy of the content according to the interest of the consumers near to this cache node in this content. Second, IoT applications impose another important requirement, which is the data security. Security in ICN is handled by the content itself rather than point to point channels. The majority of solutions proposed in the ICN context use self-certifying names to ensure security by adding the hash of the content and the key in the retrieved data itself. These methods make the content name non-human readable which contradicts some IoT requirements. We argue that the security is a very interesting direction in the ICN context. Finally, studies revolving around the NDN-based IoT networks are left with implementation lack. In fact, the majority of the researches focusing on this context are evaluated based on simulations. Real platforms implementation direction is still in its infancy. This important step may change the history of ICN, especially in IoT networks. All researchers in the ICN community are aware that the establishment of an information-centric Internet is a very harsh task since it demands to rebuild the Internet from scratch. Nevertheless, all the important innovations that the history has known have started from

a simple idea and followed a slow evolution to reach their success.

Glossary

AFRIM Adaptive Forwarding based Link Recovery for Efficient Mobility support. 96

ARMA Autoregressive Moving Average. 77

Btw Betweenness centrality. 66

CDN Content Distribution Networks. 26

COBRA COntent-driven Bloom filter based Routing Algorithm. 94

CS Content Store. 37

Edge Edge Caching. 66

FIB Forwarding Interest Base. 37

FIFO First In First Out. 51

ICN Information-Centric Networking. 17

IoT Internet of Things. 15

LCD Leave Copy Down. 66

LCE Leave Copy Everywhere. 65

LFF Least Fresh First. 85

LFU Least Frequently Used. 51

LPM Longest Prefix Match. 37

LRU Least Recently Used. 51

NDN Named Data Networking. 37

NRS Name Resolution System. 29

P2P Peep-to-Peer. 26

PIT Pending Interest Table. 37

ProbCache Probabilistic Cache. 66

RR Random Replacement. 52

Bibliography

[Ahlgren 2010] B. Ahlgren, C. Dannewitz M. D'Ambrosio, A. Eriksson, J. Golic, B. Gronvall, D. Horne, A. Lindgren, O. Mammela, M. Marchisio, J. Makela, S. Nechifor, B. Ohlman, K. Pentikousis, S. Randriamasy, T. Rautio, E. Renault, P. Seittenranta, O. Strandberg, B. Tarnauca, V. Vercellone and D. Zeghlache. *Second NetInf Architecture Description*, April 2010. Deliverable D-6.2 v2.0, 4WARD EU FP7 Project. (Cited on page 36.)

[Ahlgren 2012] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman. *A survey of information-centric networking.* IEEE Communications Magazine, vol. 50, no. 7, pages 26–36, July 2012. (Cited on pages 27 and 28.)

[Ain 2009] M. Ain, D. Trossen, P. Nikander, S. Tarkoma, K. Visala, K. Rimey, T. Burbridge, J. Rajahalme, J. Tuononen, P. Jokela, J. Kjallman, J. Ylitalo, J. Riihijarvi, B. Gajic, G. Xylomenos, P. Savolainen and D. Lagutin. *Architecture Definition, Component Descriptions, and Requirements*, Feb 2009. Deliverable D2.3, PSIRP EU FP7 Project. (Cited on page 35.)

[Al-Turjman 2013] F. M. Al-Turjman, A. E. Al-Fagih and H. S. Hassanein. *A value-based cache replacement approach for Information-Centric Networks.* In 38th Annual IEEE Conference on Local Computer Networks - Workshops, pages 874–881, Oct 2013. (Cited on page 52.)

[Alaya 2014] M. Ben Alaya, Y. Banouar, T. Monteil, C. Chassot and K. Drira. *OM2M: Extensible ETSI-compliant M2M Service Platform with Self-configuration Capability.* In Proceedings of the 5th International Conference on Ambient Systems, Networks and Technologies (ANT 2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014), Hasselt, Belgium, June 2-5, 2014, pages 1079–1086, 2014. (Cited on page 47.)

[Alimi 2013] R. Alimi, A. Rahman, D. Kutscher, Y. Yang, H. Song and K. Pentikousis. *DECoupled Application Data Enroute (DECADE).* RFC 7069, November 2013. (Cited on page 47.)

[Amadeo 2013] M. Amadeo, C. Campolo, A. Molinaro and N. Mitton. *Named Data Networking: A natural design for data collection in Wireless Sensor Networks.* In 2013 IFIP Wireless Days (WD), pages 1–6, Nov 2013. (Cited on page 47.)

[Amadeo 2014a] M. Amadeo, C. Campolo, A. Iera and A. Molinaro. *Named data networking for IoT: An architectural perspective.* In Networks and Communications (EuCNC), 2014 European Conference on, pages 1–5, June 2014. (Cited on pages 46 and 63.)

[Amadeo 2014b] M. Amadeo, C. Campolo, A. Molinaro and G. Ruggeri. *Content-centric wireless networking: A survey.* Computer Networks, vol. 72, pages 1 – 13, 2014. (Cited on pages 46 and 47.)

[Amadeo 2015] M. Amadeo, C. Campolo, A. Iera and A. Molinaro. *Information Centric Networking in IoT scenarios: The case of a smart home.* In Communications (ICC), 2015 IEEE International Conference on, pages 648–653, June 2015. (Cited on pages 17, 46 and 47.)

[Amadeo 2016a] M. Amadeo, O. Briante, C. Campolo, A. Molinaro and G. Ruggeri. *Information-centric networking for {M2M} communications: Design and deployment.* Computer Communications, pages –, 2016. (Cited on pages 46 and 47.)

[Amadeo 2016b] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar and A. V. Vasilakos. *Information-centric networking for the internet of things: challenges and opportunities.* IEEE Network, vol. 30, no. 2, pages 92–100, March 2016. (Cited on page 46.)

[Azgin 2014] A. Azgin, R. Ravindran and G. Wang. *A Scalable Mobility-Centric Architecture for Named Data Networking.* CoRR, vol. abs/1406.7049, 2014. (Cited on page 57.)

[Baccelli 2014] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt and M. Wählisch. *Information Centric Networking in the IoT: Experiments with NDN in the Wild.* CoRR, vol. abs/1406.6608, 2014. (Cited on pages 17, 33, 46 and 47.)

[Badov 2014] M. Badov, A. Seetharam, J. Kurose, V. Firoiu and S. Nanda. *Congestion-aware Caching and Search in Information-centric Networks.* In Proceedings of the 1st ACM Conference on Information-Centric Networking, ACM-ICN '14, pages 37–46, New York, NY, USA, 2014. ACM. (Cited on page 67.)

[Berners-Lee 1995] T. Berners-Lee. *Propagation, replication and caching on the web.* http://www.w3.org/pub/WWW/Propagation/Activity.html, 1995. (Cited on page 50.)

[Box 1970] G. Box and G. M. Jenkins. Time series analysis: Forecasting and control. Holden-Day Inc., San Francisco, 1st édition, 1970. (Cited on page 78.)

[Breslau 1999] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker. *Web caching and Zipf-like distributions: evidence and implications.* In INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 1, pages 126–134 vol.1, Mar 1999. (Cited on page 47.)

[Brockwell 1991] P. J. Brockwell and R. A. Davis. Time series: Theory and methods. Springer New York, 1991. (Cited on page 78.)

[Caines 1972] P. E. Caines. *Relationship between Box-Jenkins-ÃÂstrÃÂ¶m control and Kalman linear regulator.* Electrical Engineers, Proceedings of the Institution of, vol. 119, no. 5, pages 615–620, May 1972. (Cited on page 77.)

[Calvert 1997] K. L. Calvert, M. B. Doar and E. W. Zegura. *Modeling Internet topology.* IEEE Communications Magazine, vol. 35, no. 6, pages 160–163, June 1997. (Cited on page 104.)

[Camp 2002] T. Camp, J. Boleng and V. Davies. *A Survey of Mobility Models for Ad Hoc Network Research.* Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile ad hoc networking: research, trends and applications, vol. 2, pages 483–502, 2002. (Cited on page 120.)

[Chai 2012] W. K. Chai, D. He, I. Psaras and G. Pavlou. *Cache "Less for More" in Information-centric Networks.* In Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I, IFIP'12, pages 27–40. Springer-Verlag, 2012. (Cited on pages 66 and 67.)

[Chankhunthod 1996] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz and K. J. Worrell. *A Hierarchical Internet Object Cache.* In Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference, ATEC '96, pages 13–13, Berkeley, CA, USA, 1996. USENIX Association. (Cited on page 51.)

[Chiocchetti 2013a] 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, pages 9–14, New York, NY, USA, 2013. ACM. (Cited on pages 94 and 95.)

[Chiocchetti 2013b] R. Chiocchetti, D. Rossi and G. Rossini. *ccnSim: An highly scalable CCN simulator*. In Communications (ICC), 2013 IEEE International Conference on, pages 2309–2314, June 2013. (Cited on page 51.)

[Chiocchetti 2013c] R. Chiocchetti, D. Rossi and G. Rossini. *ccnSim: An highly scalable CCN simulator*. In Communications (ICC), 2013 IEEE International Conference on, pages 2309–2314, June 2013. (Cited on page 103.)

[Cho 2012] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi and S. Pack. *WAVE: Popularity-based and collaborative in-network caching for content-oriented networks*. In 2012 Proceedings IEEE INFOCOM Workshops, pages 316–321, March 2012. (Cited on page 51.)

[Cisco 1999] Cisco. *Cisco VNI Forecast and Methodology, 2015-2020*. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, 1999. (Cited on pages 15 and 26.)

[Dabirmoghaddam 2014] A. Dabirmoghaddam and J.J. Barijough M. M.and Garcia-Luna-Aceves. *Understanding Optimal Caching and Opportunistic Caching at "the Edge" of Information-centric Networks*. In Proceedings of the 1st ACM Conference on Information-Centric Networking, ACM-ICN '14, pages 47–56, New York, NY, USA, 2014. ACM. (Cited on page 68.)

[Dahlin 1994] M. D. Dahlin, R. Y. Wang, T. E. Anderson and D. A. Patterson. *Cooperative Caching: Using Remote Client Memory to Improve File System Performance*. In Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, OSDI '94, Berkeley, CA, USA, 1994. USENIX Association. (Cited on page 48.)

[Dan 2011] G. Dan. *Cache-to-Cache: Could ISPs Cooperate to Decrease Peer-to-Peer Content Distribution Costs*. IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 9, pages 1469–1482, Sept 2011. (Cited on page 48.)

[Detti 2013] A. Detti, A. Caponi, G. Tropea, G. Bianchi and N. Blefari-Melazzi. *On the interplay among naming, content validity and caching in Information Centric Networks*. In 2013 IEEE Global Communications Conference (GLOBECOM), pages 2108–2113, Dec 2013. (Cited on page 28.)

[Dingle 1996] A. Dingle and T. Pártl. *Web Cache Coherence*. Comput. Netw. ISDN Syst., vol. 28, no. 7-11, pages 907–920, May 1996. (Cited on page 49.)

[Dinh 2013] N. Dinh and Y. Kim. *Potential of information-centric wireless sensor and actor networking*. In Computing, Management and Telecommunications (ComManTel), 2013 International Conference on, pages 163–168. IEEE, January 2013. (Cited on page 47.)

[Duquennoy 2011] S. Duquennoy, F. Österlind and A. Dunkels. *Lossy Links, Low Power, High Throughput*. In Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, SenSys '11, pages 12–25, New York, NY, USA, 2011. ACM. (Cited on page 32.)

[Eum 2012] S. Eum, K. Nakauchi, M. Murata, Y. Shoji and N. Nishinaga. *CATT: Potential Based Routing with Content Caching for ICN*. In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, ICN '12, pages 49–54, New York, NY, USA, 2012. ACM. (Cited on page 67.)

[Fan 2000] L. Fan, P. Cao, J. Almeida and A. Z. Broder. *Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol*. IEEE/ACM Trans. Netw., vol. 8, no. 3, pages 281–293, June 2000. (Cited on page 48.)

[Fayazbakhsh 2013] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K.C. Ng, V. Sekar and S. Shenker. *Less Pain, Most of the Gain: Incrementally Deployable ICN*. SIGCOMM Comput. Commun. Rev., vol. 43, no. 4, pages 147–158, August 2013. (Cited on pages 66, 68, 107 and 111.)

[Feldmann 2007] A. Feldmann. *Internet Clean-slate Design: What and Why?* SIGCOMM Comput. Commun. Rev., vol. 37, no. 3, pages 59–64, July 2007. (Cited on pages 26 and 27.)

[FP7 2010a] FP7. *COMET project*. http://www.comet-project.org/, 2010. (Cited on pages 36 and 41.)

[FP7 2010b] FP7. *CONVERGENCE project*. http://www.ict-convergence.eu/, 2010. (Cited on pages 37 and 41.)

[FP7 2010c] FP7. *PURSUIT project*. http://www.fp7-pursuit.eu/PursuitWeb/, 2010. (Cited on pages 35 and 41.)

[FP7 2010d] FP7. *SAIL project*. http://www.sail-project.eu/, 2010. (Cited on pages 36 and 41.)

[Francois 2013] J. Francois, T. Cholez and T. Engel. *CCN traffic optimization for IoT*. In Network of the Future (NOF), 2013 Fourth International Conference on the, pages 1–5, Oct 2013. (Cited on page 46.)

[Fricker 2012] C. Fricker, P. Robert, J. Roberts and N. Sbihi. *Impact of traffic mix on caching performance in a content-centric network*. In IEEE NOMEN 2012, Workshop on Emerging Design Choices in Name-Oriented Networking, Orlando, USA, March 2012. (Cited on page 66.)

[Gubbi 2013] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami. *Internet of Things (IoT): A vision, architectural elements, and future directions*. Future Generation Computer Systems, vol. 29, no. 7, pages 1645 – 1660, 2013. (Cited on page 17.)

[Hail 2015] M.A. Hail, M. Amadeo, A. Molinaro and S. Fischer. *Caching in Named Data Networking for the wireless Internet of Things*. In Recent Advances in Internet of Things (RIoT), 2015 International Conference on, pages 1–6, April 2015. (Cited on pages 17, 46, 49, 50 and 107.)

[Han 2014] D. Han, M. Lee, K. Cho, T. Kwon and Y. Choi. *Publisher mobility support in content centric networks*. In The International Conference on Information Networking 2014 (ICOIN2014), pages 214–219, Feb 2014. (Cited on page 55.)

[Heidemann 2001] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan. *Building Efficient Wireless Sensor Networks with Low-level Naming*. SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pages 146–159, October 2001. (Cited on pages 17 and 46.)

[Hermans 2011] F. Hermans, E. Ngai and P. Gunningberg. *Mobile sources in an information-centric network with hierarchical names: An indirection approach*. In in SNCNW, 2011. (Cited on page 55.)

[Hermans 2012] F. Hermans, E. Ngai and P. Gunningberg. *Global Source Mobility in the Content-centric Networking Architecture*. In Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications, NoM '12, pages 13–18, New York, NY, USA, 2012. ACM. (Cited on page 57.)

[Hoque 2013] A. K. M. M. 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, ICN '13, pages 15–20, New York, NY, USA, 2013. ACM. (Cited on page 94.)

[Hyndman 2014] R.J. Hyndman and G. Athanasopoulos. Forecasting: principles and practice:. OTexts, 2014. (Cited on page 77.)

[Imbrenda 2014] C. Imbrenda, L. Muscariello and D. Rossi. *Analyzing Cacheable Traffic in Isp Access Networks for Micro Cdn Applications via Content-centric Networking*. In Proceedings of the 1st ACM Conference on Information-Centric Networking, ACM-ICN '14, pages 57–66, New York, NY, USA, 2014. ACM. (Cited on page 68.)

[Ishaq 2013] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman and P. Demeester. *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*. Journal of Sensor and Actuator Networks, vol. 2, no. 2, pages 235–287, 2013. (Cited on page 32.)

[Jacobson 2009] 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, pages 1–12. ACM, 2009. (Cited on pages 17, 26, 27, 37 and 65.)

[Katsaros 2011] K. Katsaros, G. Xylomenos and G. C. Polyzos. *MultiCache: An overlay architecture for information-centric networking*. Computer Networks, vol. 55, no. 4, pages 936 – 947, 2011. Special Issue on Architectures and Protocols for the Future Internet. (Cited on page 48.)

[Katsaros 2014] K. Katsaros, W. Chai, N. Wang, G. Pavlou, H. Bontius and M. Paolone. *Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications*. Network, IEEE, vol. 28, no. 3, pages 58–64, May 2014. (Cited on page 47.)

[Kazar 1988] M. L. Kazar. *Synchronization and caching issues in the Andrew file system*. In USENIX, pages 27–36, 1988. (Cited on page 50.)

[Kim 2012] D. Kim, J. Kim, Y. Kim, H. Yoon and I. Yeom. *Mobility Support in Content Centric Networks*. In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, ICN '12, pages 13–18, New York, NY, USA, 2012. ACM. (Cited on page 55.)

[Kim 2015] D. Kim, J. Kim, Y. Kim, H. Yoon and I. Yeom. *End-to-end Mobility Support in Content Centric Networks*. Int. J. Commun. Syst., vol. 28, no. 6, pages 1151–1167, April 2015. (Cited on page 55.)

[Koponen 2007] T. Koponen, A. Ermolinskiy, M. Chawla, K. H. Kim, I. Stoica, B. Chun and S. Shenker. *A data-oriented (and beyond) network architecture.* In In SIGCOMM, 2007. (Cited on pages 35 and 41.)

[Kurita 2017] T. Kurita, I. Sato, K. Fukuda and T. Tsuda. *An extension of Information-Centric Networking for IoT applications.* In 2017 International Conference on Computing, Networking and Communications (ICNC), pages 237–243, Jan 2017. (Cited on page 46.)

[Kutscher 2011] D. Kutscher, G. Morabito, I. Solis, B. Ohlman, G. C. Polyzos and L. Zhang, editeurs. 2011 ACM SIGCOMM workshop on information-centric networking, ICN 2011, toronto, on, canada, august 19, 2011. ACM, 2011. (Cited on page 66.)

[LAAS-CNRS 2013] LAAS-CNRS. *ADREAM.* http://www.laas.fr/1-32329-Le-batiment-intelligent-Adream-instrumente-et-econome-en-energie.php, 2013. (Cited on pages 78 and 106.)

[Laoutaris 2004] N. Laoutaris, S. Syntila and I. Stavrakakis. *Meta algorithms for hierarchical Web caches.* In IEEE International Conference on Performance, Computing, and Communications, 2004, pages 445–452, 2004. (Cited on page 65.)

[Laoutaris 2006] N. Laoutaris, H. Che and I. Stavrakakis. *The {LCD} interconnection of {LRU} caches and its analysis.* Performance Evaluation, vol. 63, no. 7, pages 609 – 634, 2006. (Cited on page 66.)

[Lee 2011] M. Lee, K. Cho, K. Park, T. Kwon and Y. Choi. *SCAN: Scalable Content Routing for Content-Aware Networking.* In 2011 IEEE International Conference on Communications (ICC), pages 1–5, June 2011. (Cited on pages 94 and 95.)

[Lee 2012] J. Lee, S. Cho and D. Kim. *Device mobility management in content-centric networking.* IEEE Communications Magazine, vol. 50, no. 12, pages 28–34, December 2012. (Cited on page 55.)

[Lindgren 2016] A. Lindgren, F. B. Abdesslem, B. Ahlgren, O. Schelén and A. M. Malik. *Design choices for the IoT in Information-Centric Networks.* In 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC), pages 882–888, Jan 2016. (Cited on page 46.)

[Liu 2011] R. Liu, W. Wu, H. Zhu and D. Yang. *M2M-Oriented QoS Categorization in Cellular Network*. In Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on, pages 1–5, Sept 2011. (Cited on page 76.)

[Liu 2012] H. Liu, X. De Foy and D. Zhang. *A Multi-level DHT Routing Framework with Aggregation*. In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, ICN '12, pages 43–48, New York, NY, USA, 2012. ACM. (Cited on pages 94 and 95.)

[Makridakis 1997] S. Makridakis and M. Hibon. *ARMA Models and the Box-Jenkins Methodology*. Journal of Forecasting, vol. 16, no. 3, pages 147–163, 1997. (Cited on page 78.)

[Meddeb 2014] M. Meddeb, M. Ben Alaya, T. Monteil, A. Dhraief and K. Drira. *M2M Platform with Autonomic Device Management Service*. Procedia Computer Science, vol. 32, no. 0, pages 1063 – 1070, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014). (Cited on page 16.)

[Meddeb 2015] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *Cache coherence in Machine-to-Machine Information Centric Networks*. In 40th IEEE Conference on Local Computer Networks, LCN 2015, Clearwater Beach, FL, USA, October 26-29, 2015, pages 430–433, 2015. (Cited on page 19.)

[Meddeb 2017a] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *AFIRM: Adaptive Forwarding based Link Recovery for Mobility Support in NDN-based IoT networks*. Submitted to Future Generation Computer Systems, 2017. (Cited on page 19.)

[Meddeb 2017b] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *Cache freshness in Named Data Networking of Internet of things*. Under review in The computer Journal with minor revision, 2017. (Cited on page 19.)

[Meddeb 2017c] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *How to cache in ICN/IoT networks?* 14th ACS/IEEE International Conference on Computer Systems and Applications, 2017. (Cited on page 19.)

[Meddeb 2017d] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *The Least Fresh First Cache Replacement Policy for NDN/IoT Networks.* submitted to 25th IEEE International Conference on Network Protocols, 2017. (Cited on page 19.)

[Meddeb 2017e] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira. *Named Data Networking: A promising architecture for the Internet of things (IoT).* Journal on Semantic Web and Information Systems, 2017. (Cited on page 19.)

[Michel 1998] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd and V. Jacobson. *Adaptive web caching: towards a new global caching architecture.* Computer Networks and {ISDN} Systems, vol. 30, no. 22–23, pages 2169 – 2177, 1998. (Cited on page 48.)

[Minerva 2015] R. Minerva, A. Biru and D. Rotondi. *Towards a Definition of the Internet of Things (IoT).* In IEEE Internet Initiative, May 2015. (Cited on page 48.)

[Montavont 2014] J. Montavont, D. Roth and T. Noel. *Mobile {IPv6} in Internet of Things: Analysis, experimentations and optimizations.* Ad Hoc Networks, vol. 14, pages 15 – 25, 2014. (Cited on page 32.)

[Msadaa 2016] I. C. Msadaa and A. Dhraief. *1 - Internet of Things in Support of Public Safety Networks: Opportunities and Challenges.* In D. Camara and N. Nikaein, editeurs, Wireless Public Safety Networks 2, pages 1 – 23. Elsevier, 2016. (Cited on pages 32 and 33.)

[NSF 2010a] NSF. *Mobility First project.* http://mobilityfirst.winlab.rutgers.edu/, 2010. (Cited on pages 38 and 41.)

[NSF 2010b] NSF. *Named Data Networking project.* http://www.named-data.net/, 2010. (Cited on pages 37 and 41.)

[Panigrahi 2014] B. Panigrahi, S. Shailendra, H. K. Rath and A. Simha. *Universal caching model and Markov-based cache analysis for information centric networks.* In 2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pages 1–6, Dec 2014. (Cited on page 52.)

[Pathan 2007] A. Pathan and R. Buyya. *A taxonomy and survey of content delivery networks,.* Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia., Feb. 2007. (Cited on page 26.)

[Pavlou 2013] G. Pavlou. *Information-Centric Networking and In-Network Cache Management: Overview, Trends and Challenges.* In Keynote speech of the 9th IFIP/IEEE Conference on Network and Service Management, 2013. (Cited on page 48.)

[Pentikousis 2015] K. Pentikousis, B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. B. Davies, A. Molinaro and S. Eum. *Information-Centric Networking: Baseline Scenarios.* RFC 7476, March 2015. (Cited on pages 17 and 46.)

[Pentikousis 2016] K. Pentikousis, B. Ohlman, E. B. Davies, G. Boggia and S. Spirou. *Information-Centric Networking: Evaluation and Security Considerations.* RFC 7945, September 2016. (Cited on page 104.)

[Podlipnig 2003] S. Podlipnig and L. Böszörmenyi. *A Survey of Web Cache Replacement Strategies.* ACM Comput. Surv., vol. 35, no. 4, pages 374–398, December 2003. (Cited on page 51.)

[Psaras 2012] I. Psaras, W. K. Chai and G. Pavlou. *Probabilistic In-network Caching for Information-centric Networks.* In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, ICN '12, pages 55–60, New York, NY, USA, 2012. ACM. (Cited on pages 51 and 66.)

[Psaras 2014] I. Psaras, W. K. Chai and G. Pavlou. *In-Network Cache Management and Resource Allocation for Information-Centric Networks.* IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 11, pages 2920–2931, Nov 2014. (Cited on page 67.)

[Quevedo 2014a] J. Quevedo, D. Corujo and R. Aguiar. *A case for ICN usage in IoT environments.* In Global Communications Conference (GLOBECOM), 2014 IEEE, pages 2770–2775, Dec 2014. (Cited on pages 17 and 48.)

[Quevedo 2014b] J. Quevedo, D. Corujo and R. Aguiar. *Consumer driven information freshness approach for content centric networking.* In Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on, pages 482–487, April 2014. (Cited on pages 17, 50 and 107.)

[Rao 2014] Y. Rao, H. Luo, D. Gao, H. Zhou and H. Zhang. *LBMA: A novel Locator Based Mobility support Approach in Named Data Networking.* China Communications, vol. 11, no. 4, pages 111–120, April 2014. (Cited on page 57.)

[Ravindran 2012] R. Ravindran, S. Lo, X. Zhang and G. Wang. *Supporting seamless mobility in named data networking.* In 2012 IEEE International Conference on Communications (ICC), pages 5854–5869, June 2012. (Cited on page 57.)

[Ren 2013] Z. Ren, M.A. Hail and H. Hellbruck. *CCN-WSN - A lightweight, flexible Content-Centric Networking protocol for wireless sensor networks.* In Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on, pages 123–128, April 2013. (Cited on page 47.)

[Rosensweig 2010] E. J. Rosensweig, J. Kurose and D. Towsley. *Approximate Models for General Cache Networks.* In 2010 Proceedings IEEE INFOCOM, pages 1–9, March 2010. (Cited on page 48.)

[Rossi 2011] D. Rossi and G. Rossini. *Caching performance of content centric networks under multi-path routing.* Rapport technique, Telecom ParisTech, 2011. (Cited on page 107.)

[Rossi 2012] D. Rossi and G. Rossini. *On sizing CCN content stores by exploiting topological information.* In 2012 Proceedings IEEE INFOCOM Workshops, pages 280–285, March 2012. (Cited on page 66.)

[Rossini 2012] G. Rossini and D. Rossi. *A dive into the caching performance of Content Centric Networking.* In CAMAD, pages 105–109. IEEE, 2012. (Cited on page 66.)

[Saleh 2006] O. Saleh and M. Hefeeda. *Modeling and Caching of Peer-to-Peer Traffic.* In Proceedings of the 2006 IEEE International Conference on Network Protocols, pages 249–258, Nov 2006. (Cited on page 47.)

[Sandberg 1985] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh and B. Lyon. *Design and implementation of the Sun Network filesystem.* In USENIX, Summer 1985. (Cited on page 50.)

[Sourlas 2014] V. Sourlas, P. Flegkas and L. Tassiulas. *A novel cache aware routing scheme for Information-Centric Networks.* Computer Networks, vol. 59, pages 44 – 61, 2014. (Cited on page 51.)

[Teubler 2013] T. Teubler, M. A. Hail and H. Hellbruck. *Efficient Data Aggregation with CCNx in Wireless Sensor Networks.* In EUNICE, volume 8115 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2013. (Cited on page 47.)

[Tortelli 2014] M. Tortelli, L. A. Grieco, G. Boggia and K. Pentikousisy. *COBRA: Lean intra-domain routing in NDN.* In 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), pages 839–844, Jan 2014. (Cited on pages 94 and 95.)

[Tsilopoulos 2011] C. Tsilopoulos and G. Xylomenos. *Supporting Diverse Traffic Types in Information Centric Networks.* In Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking, ICN '11, pages 13–18, New York, NY, USA, 2011. ACM. (Cited on page 67.)

[Vasilakos 2012] X. Vasilakos, V. A. Siris, G. C. Polyzos and M. Pomonis. *Proactive Selective Neighbor Caching for Enhancing Mobility Support in Information-centric Networks.* In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, ICN '12, pages 61–66, New York, NY, USA, 2012. ACM. (Cited on pages 70, 71 and 72.)

[Vleeschauwer 2011] D. De Vleeschauwer and D. C. Robinson. *Optimum caching strategies for a telco CDN.* Bell Labs Technical Journal, vol. 16, no. 2, pages 115–132, Sept 2011. (Cited on page 47.)

[Vural 2014] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong and R. Tafazolli. *In-network caching of Internet-of-Things data.* In Communications (ICC), 2014 IEEE International Conference on, pages 3185–3190, June 2014. (Cited on page 48.)

[Vural 2016] S. Vural, N. Wang, P. Navaratnam and R. Tafazolli. *Caching Transient Data in Internet Content Routers.* IEEE/ACM Transactions on Networking, vol. PP, no. 99, pages 1–14;, 2016. (Cited on page 49.)

[W3C 1996] World Wide Web Consortium W3C. *W3C httpd.* http://www.w3.org/pub/WWW/Daemon/, 1996. (Cited on page 50.)

[W3C 2006] World Wide Web Consortium W3C. *Hypertext Transfer Protocol.* http://www.w3.org/pub/WWW/Protocols/, 2006. (Cited on page 50.)

[Wang 2012a] L. Wang, A. K. M. M. Hoque, C. Yi, A. Alyyan and B. Zhang. *OSPFN: An OSPF Based Routing Protocol for Named Data Networking.* 2012. (Cited on page 94.)

[Wang 2012b] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru and L. Zhang. *Data Naming in Vehicle-to-Vehicle Communications.* In In Proceedings of INFOCOM 2012

Workshop on Emerging Design Choices in Name-Oriented Networking, March 2012. (Cited on page 47.)

[Wang 2012c] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee and S. Yang. *Advertising cached contents in the control plane: Necessity and feasibility.* In 2012 Proceedings IEEE INFOCOM Workshops, pages 286–291, March 2012. (Cited on pages 94 and 95.)

[Wang 2013a] J. M. Wang, J. Zhang and B. Bensaou. *Intra-AS Cooperative Caching for Content-centric Networks.* In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking, ICN '13, pages 61–66, New York, NY, USA, 2013. ACM. (Cited on pages 48 and 67.)

[Wang 2013b] L. Wang, O. Waltari and J. Kangasharju. *MobiCCN: Mobility support with greedy routing in Content-Centric Networks.* In 2013 IEEE Global Communications Conference (GLOBECOM), pages 2069–2075, Dec 2013. (Cited on pages 55 and 57.)

[Wong 2006] K. Wong. *Web cache replacement policies: a pragmatic approach.* IEEE Network, vol. 20, no. 1, pages 28–34, Jan 2006. (Cited on page 51.)

[Xu 2014] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu and F. Bu. *Ubiquitous Data Accessing Method in IoT-Based Information System for Emergency Medical Services.* IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pages 1578–1586, May 2014. (Cited on page 46.)

[Xylomenos 2012] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V.A. Siris and G.C. Polyzos. *Caching and mobility support in a publish-subscribe internet architecture.* Communications Magazine, IEEE, vol. 50, no. 7, pages 52–58, July 2012. (Cited on page 64.)

[Xylomenos 2014] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros and G. C. Polyzos. *A Survey of Information-Centric Networking Research.* IEEE Communications Surveys Tutorials, vol. 16, no. 2, pages 1024–1049, Second 2014. (Cited on pages 17, 27 and 28.)

[Yi 2012] C. Yi, A. Afanasyev, L. Wang, B. Zhang and L. Zhang. *Adaptive forwarding in named data networking.* Computer Communication Review, vol. 42, pages 62–67, 2012. (Cited on page 93.)

[Yi 2013] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang and L. Zhang. *A case for stateful forwarding plane*. Computer Communications, vol. 36, no. 7, pages 779 – 791, 2013. (Cited on page 92.)

[Yi 2014] 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 ACM Conference on Information-Centric Networking (ICN'2014), September 2014. (Cited on page 92.)

[Zhang 2010] L. Zhang, D. Estrin, J. Bruke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley and E. Yeh. *Named Data Networking (NDN) Project*. October 2010. (Cited on page 65.)

[Zhang 2013] G. Zhang, Y. Li and T. Lin. *Caching in information centric networking: A survey*. Computer Networks, vol. 57, no. 16, pages 3128 – 3141, 2013. Information Centric Networking. (Cited on pages 30, 47 and 65.)

[Zhang 2016] Y. Zhang, D. Raychadhuri, L. A. Grieco, E. Baccelli, J. Burke, R. Ravindran and G. Wang. *ICN based Architecture for IoT - Requirements and Challenges*. Internet-draft, Internet Research Task Force (IRTF), 2016. version 2 expires on February 29. (Cited on pages 32 and 46.)

[Zhou 2014] Z. Zhou, X. Tan, H. Li, Z. Zhao and D. Ma. *MobiNDN: A mobility support architecture for NDN*. In Proceedings of the 33rd Chinese Control Conference, pages 5515–5520, July 2014. (Cited on pages 31 and 53.)

[Zhu 2013] Z. Zhu, A. Afanasyev and L. Zhang. *A New Perspective on Mobility Support*. Technical Report NDN-0013, NDN, July 2013. (Cited on page 55.)