Universidade de Aveiro
2019

Departamento de
Electrónica, Telecomunicações e Informática

Carlos Eduardo Magalhães Guimarães

Interoperabilidade e Mobilidade na Internet do Futuro

Interoperability and Mobility in the Future Internet

Universidade de Aveiro
2019

Departamento de
Electrónica, Telecomunicações e Informática

Carlos Eduardo
Magalhães Guimarães

Interoperabilidade e Mobilidade na Internet do Futuro

Interoperability and Mobility in the Future Internet

Tese apresentada às Universidades de Minho, Aveiro e Porto para cumprimento dos requisitos necessários à obtenção do grau de Doutor no âmbito do programa doutoral conjunto MAP-i, realizada sob a orientação científica do Professor Doutor Rui Luis Andrade Aguiar, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Daniel Nunes Corujo, Investigador Doutorado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Apoio financeiro da Fundação para a Ciência e Tecnologia (FCT) através da bolsa SFRH/BD/96553/2013.

o júri / the jury

presidente / president	**Doutor Luís António Ferreira Martins Dias Carlos** Professor Catedrático, Universidade de Aveiro
vogais / examiners committee	**Doutor Ivan Vidal Fernández** Professor Visitante, Universidad Carlos III de Madrid
	Doutor Paulo Alexandre Ferreira Simões Professor Auxiliar, Universidade de Coimbra
	Doutor Paulo Martins de Carvalho Professor Associado, Universidade do Minho
	Doutor Amaro Fernandes de Sousa Professor Auxiliar, Universidade de Aveiro
	Doutor Rui Luís Andrade Aguiar (orientador) Professor Catedrático, Universidade de Aveiro

agradecimentos / acknowledgements

Agradeço ao meus orientadores, Prof. Dr. Rui L. Aguiar e Dr. Daniel Corujo, pela supervisão deste trabalho, cujo resultado não seria possível sem a sua colaboração.

Agradeço ao Instituto de Telecomunicações que me proporcionou todas as condições necessárias para o desenvolvimento deste trabalho. Em particular agradeço ao grupo de investigação "Telecommunications and Networking - Av" do Instituto de Telecomunicações - Pólo de Aveiro.

Agradeço à Fundação para a Ciência e Tecnologia (FCT) pela bolsa de doutoramento (SFRH/BD/96553/2013) que me foi atribuída.

Agradeço à minha família e amigos que me acompanharam ao longo destes anos.

Palavras Chave

Internet do Futuro; Interoperabilidade; Mobilidade; Redes Centradas na Informação; Redes Definidas por Software;

Resumo

A Internet do Futuro tem sido alvo de vários estudos nos últimos anos, com a proposta de arquitecturas de rede seguindo quer abordagens evolutionárias (por exemplo, Redes Definidas por Software (SDN)) quer abordagens disruptivas (por exemplo, Redes Centradas na Informação (ICN)). Cada uma destas arquitecturas de rede visa providenciar melhores soluções relativamente a determinados requisitos de utilização da Internet e, portanto, uma Internet do Futuro heterogénea composta por diversas arquitecturas de rede torna-se uma possibilidade, onde cada uma delas é usada para optimizar diferentes casos de utilização. Para além disso, o aumento do número de dispositivos móveis, com especificações acrescidas e com suporte para diferentes tecnologias de conectividade, está a mudar os padrões do tráfego na Internet.
Assim, esta tese foca-se no estudo de aspectos de interoperabilidade e mobilidade em arquitecturas de rede da Internet do Futuro, dois importantes requisitos que necessitam de ser satisfeitos para que a adopção destas arquitecturas de rede seja considerada. A primeira contribuição desta tese é uma solução de interoperabilidade que, uma vez que permite que recursos possam ser partilhados por diferentes arquitecturas de rede, evita que os recursos estejam restringidos a uma determinada arquitectura de rede e, ao mesmo tempo, promove a adopção de novas arquitecturas de rede. A segunda contribuição desta tese consiste no desenvolvimento de extensões para arquitecturas de rede baseadas em SDN ou ICN através dos mecanismos propostos na norma IEEE 802.21 com o objectivo de facilitar e optimizar os processos de mobilidade nessas arquitecturas de rede. Finalmente, a terceira contribuição desta tese é a definição de uma solução de mobilidade envolvendo diferentes arquitecturas de rede que permite a mobilidade de dispositivos móveis entre redes de acesso que suportam diferentes arquitecturas de rede sem que estes percam o acesso aos recursos que estão a ser acedidos. Todas as soluções propostas foram avaliadas com os resultados a demonstrar a viabilidade de cada uma das soluções e o impacto que têm na comunicação.

Keywords Future Internet; Interoperability; Mobility; Information Centric Networking; Software Defined Networking;

Abstract

Research on Future Internet has been gaining traction in recent years, with both evolutionary (e.g., Software Defined Networking (SDN)-based architectures) and clean-slate network architectures (e.g., Information Centric Networking (ICN) architectures) being proposed. With each network architectural proposal aiming to provide better solutions for specific Internet utilization requirements, an heterogeneous Future Internet composed by several architectures can be expected, each targeting and optimizing different use case scenarios. Moreover, the increasing number of mobile devices, with increasing capabilities and supporting different connectivity technologies, are changing the patterns of traffic exchanged in the Internet.

As such, this thesis focuses on the study of interoperability and mobility in Future Internet architectures, two key requirements that need to be addressed for the widely adoption of these network architectures. The first contribution of this thesis is an interoperability framework that, by enabling resources to be shared among different network architectures, avoids resources to be restricted to a given network architecture and, at the same time, promotes the initial roll out of new network architectures. The second contribution of this thesis consists on the development of enhancements for SDN-based and ICN network architectures through IEEE 802.21 mechanisms to facilitate and optimize the handover procedures on those architectures. The last contribution of this thesis is the definition of an inter-network architecture mobility framework that enables MNs to move across access network supporting different network architectures without losing the reachability to resources being accessed. All the proposed solutions were evaluated with results highlighting the feasibility of such solutions and the impact on the overall communication.

Contents

List of Figures

List of Tables

List of Acronyms

AFTR Address Family Transition Router

AP Access Point

API Application Programming Interface

BIA Bump-in-the-API

BIH Bump-in-the-Host

BIS Bump-in-the-Stack

CCN Content Centric Networking

CDN Content Delivery Network

CN Correspondent Node

CoAP Constrained Application Protocol

CPE Customer-Premises Equipment

CPU Central Processing Unit

CS Content Store

CSS Cascading Style Sheets

DHCP Dynamic Host Configuration Protocol

DHT Distributed Hash Table

DMM Distributed Mobility Management

DNS Domain Name System

DS Dual-Stack

DTS Domain Title Service

DTSA Domain Title Service Agents

ETArch Entity Title Architecture

ETCP Entity Title Control Protocol

FA Foreign Agent

FI Future Internet

FIB Forwarding Information Base

FIC Future Internet Controller

FIFu Future Internet Fusion

FIRE Future Internet Research and Experimentation

FIXA Future Internet eXchange Anchor

FIXP Future Internet eXchange Point

FMIP Fast Handovers in Mobile IPv6

FTP File Transfer Protocol

GENI Global Environment for Network Innovations

HA Home Agent

HLS HTTP-Level Streaming

HMIP Hierarchical Mobile IPv6

HO Handover

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

ICN Information-Centric Networking

ICNRG Information-Centric Networking Research Group

ICT Information and Communication Technologies

ID Identifier

IETF Internet Engineering Task Force

IP Internet Protocol

IRTF Internet Research Task Force

ISATAP Intra-Site Automatic Tunnel Addressing Protocol

IXP Internet eXchange Point

KPI Key Performance Indicator

LIPSIN Line Speed Publish/Subscribe Inter-Networking

LMA Localized Mobility Anchor

LTE Long-Term Evolution

MAC Media Access Control

MAG Mobile Access Gateway

MICS Media Independent Control Service

MIES Media Independent Event Service

MIH Media Independent Protocol

MIHF Media Independent Handover Function

MIIS Media Independent Information Service

MIP Mobile IP

MN Mobile Node

MTU Maximum Transmission Unit

NACK Negative Acknowledgement

NAP Network Attachment Point

NAPT Network Address Port Translation

NAT Network Address Translation

NCP Network Control Program

NDN Named Data Networking

NETINF Network of Information

NIC Network Interface Controller

NR Network Resource

OS Operating System

PIT Pending Information Table

PMIP Proxy Mobile IP

PNG Portable Network Graphics

PSIRP Publish-Subscribe Internet Routing Paradigm

PURSUIT Publish Subscribe Internet Technology

RBA Resource Binding Acknowledge

RBP Resource Binding Procedure

RBU Resource Binding Update

RFC Request for Comments

RSSI Received Signal Strength Indicator

RTP Real-time Transport Protocol

RTSP Real Time Streaming Protocol

RTT Round Trip Time

SAP Service Access Point

SDN Software Defined Networking

SIIT Stateless IP/ICMP Translation Algorithm

SOCKS Socket Secure

TCP Transmission Control Protocol

TRT Transport Relay Translator

UDP User Datagram Protocol

URI Uniform Resource Identifiers

WiMAX Worldwide Interoperability for Microwave Access

WLAN Wireless Local Area Network

Chapter 1

Introduction

This introductory chapter sets the main motivation behind the work realized in this thesis, highlighting the challenges and objectives that guided its evolution. The main contributions that resulted from the developed work are then presented, finalizing with the description of the document structure, where a summary of the content of the upcoming chapters is presented.

1.1 Motivation

The Internet, representing one of mankind's most significant technological achievements, is a computer network that has been influencing the daily life of society, with a great impact on how people communicate and interact with the world in their surroundings. It has evolved, from a network of a few machines conceived for exchanging short text messages, to a worldwide platform that interconnects millions of hosts and where different kinds of services are provided. A simple interaction using voice, video and/or text communications with someone that can be right on our side or someone else on the other side of the globe, the access to different types of published contents anytime and anywhere, or even the monitoring and control of a wide range and variety of sensing devices, are just a few examples of the types of services currently enabled by the Internet.

One of the key aspects responsible for its integration into an increasing number of societal areas worldwide has been the flexibility and simplicity of the underlying protocol - the Internet Protocol (IP). However, that same success has been continuously imposing new utilization requirements which not only put that simplicity to the test, but actually hinder its own base operation as well. For example, scalable content distribution, Quality of Service (QoS), security and mobility are behaviors that are commonly granted today, but were not present at the inception of the IP protocol.

To face the continuous set of requirements that are placed over the underlying architecture, the evolution of the Internet architecture has been following an incremental strategy, where architectural adjustments address the necessary enhancements towards its enablement for future utilization scenarios, while maintaining legacy support. Such evolutionary strategy led to a vicious cycle of enhancements deployed over the base functionality of the Internet architecture, which is, at each evolution cycle, contributing to the ossification of core protocols of the Internet architecture, hindering further evolutions [56]. An example is the increasing deployment of middleboxes (e.g., firewalls, Network Address Translation (NAT) and proxies) [21], that despite being deployed for performance or security reasons, have also impacted the evolution of existing protocols such as IP [40] and Transmission Control Protocol (TCP) [57], as well as have contributed to the opaqueness of the end-to-end principle. Current advances realized under this evolutionary approach have been focusing on the continuity of the IP protocol, enhancing the current architecture with mechanisms that operate parallel to it, such as the application of Software Defined Networking (SDN) control and network visualization, as manifested in the upcoming 5th generation (5G) of telecommunication networks.

As an alternative to this incremental evolution of the Internet, clean-slate network architectures solutions started to be experimented towards the re-design of its foundation [37]. These architectures aim to better address current (and expected future) usage requirements without being restrained by existing constraints, protocols and architecture of the current deployed Internet environment [84]. One common aspect of most proposed clean-slate network architectures is that they move away from the host-centric and end-to-end principles of the IP protocol, and instead center their design in other networking elements (e.g., content, mobility and/or users). For instance, Information Centric Networking (ICN) [110] is one of such architectures, where content has a central role in the network protocol operation. The requirement to meet scalable content delivery with intrinsic security aspects motivated this shift from the current host-centric to an information-centric view where information becomes decoupled from its location.

Still, the capability of such architectures to address the current requirements of the Internet is crucial for being considered as possible candidates for a Future Internet architecture. In an increasingly mobile environment, where a multitude of mobile nodes (MNs) access the Internet, mobility becomes a preponderant requirement that needs to be fully supported by Future Internet architectures. It is therefore important to study the means through which both evolutionary and clean-slate architectures can face different usage scenarios. This is needed not only to study their potential, feasibility and practicability but also to pave the way for improvements and maturation of such architectures, that could culminate on their adoption. In particular, the adoption and rolling out of a new network architecture will depend on how it is able to interoperate with the existing networking environment.

1.2 Research Challenges and Objectives

The plethora of ways of using the Internet is continuously changing in ways that were not present at the conception of the IP protocol, leveraged by the introduction of new technologies, services and applications, A change on the originally envisioned usage patterns is being witnessed, moving from an Internet of hosts to an Internet of content (e.g., according to [25], by 2021 about 82% of the total exchanged IP traffic will be related to video traffic) and nodes are becoming mobile instead of just being stationary nodes (e.g., according to [25], by 2021 more than 63% of total IP traffic will be generated by wireless and mobile nodes). As a result, a new set of requirements are placed over the Internet architecture, with scalable content distribution and mobility the most prominent.

To cope with the new set of requirements, a set of novel network architectures for the Future Internet are being proposed, following both evolutionary and clean-slate strategies. In contrast to evolutionary approaches, which deployment is expected to be backwards-compatible with the existing networking environment (and, therefore, its deployment may prove to be simpler and more straightforward), the deployment of clean-slate architectures is not envisioned as an easy process, since they may require architectural changes ranging from the network up to the application layer and may be dependent on the agreement of different stakeholders and costs of deployment. As such, the deployment of these architectures is expected to be incrementally performed in parallel with the existing architectural environment, leading to a period of time where different network architectures need to coexist and share information with one another. The trend towards "vertical-oriented" slices in future 5G networks further reinforces the probability of this scenario [89, 90]. An example of this deployment strategy can be witnessed with the deployment of IPv6, which is being incrementally done over several years, coexisting and interoperating with IPv4. As such, in a network landscape composed by multiple parallel network architectures, the nonexistence of mechanisms that allow different architectures to exchange information may lead to the creation of information silos, where information is only available in a given network architecture. Interoperability mechanisms to allow entities in different network architectures to share information with one another are required to avoid the restriction of information to a given network architecture. At the same time, such mechanisms may promote the rolling out of new network architectures as well as their incremental adoption. As a result, the first objective to be approached by this thesis can be formulated as:

- **Objective 1:** Design generic and holistic interoperability mechanisms that allow resources deployed in different Future Internet network architectures to be accessed by users independently of its access network architecture.

The increasing proliferation of mobile devices (e.g., smartphones) with increased processing and connecting capabilities (i.e., equipped with multiple network interfaces of different technologies, including cable, wireless or cellular access) facilitates the access to the Internet while on the move and from any place. Such heterogeneity regarding the connecting capabilities opens space for a new set of opportunities for exploiting the simultaneous availability of different access technologies to ensure optimal connectivity. In other words, an always best connected experience may be provided to users taking into consideration the type of network traffic being generated and/or consumed and the network usage at each moment. However, it creates a complex heterogeneous environment where connectivity for different access networks needs to be provided. With mobility being one of the key requirements that Future Internet architectures need to address, supporting mobility mechanisms in those architectures (both evolutionary and clean-slate network architectures) may still be required in order to optimize different key performance indicators (KPIs). Leveraging context information from both the user's device (e.g., available interfaces, current link condition and surrounding networks) and network entities (e.g., number of connected devices or traffic load on each available interface) can help to decide whether to handover a given device or part of its flows to another point of attachment in order to optimize the overall operation of the network. As a result, the second objective to be approached by this thesis can be formulated as:

- **Objective 2:** Design context-aware mobility management mechanisms that allow controlling entities in both the mobile node and network entities to preemptively detect the need for an handover, and to control and manage the handover to the best available point of attachment independently of the underlying wireless technology.

The edge of the network may be an expected location for the initial roll out of native deployments of new Future Internet network architectures, by means of isolated network architectural islands interconnected with one another, either through dedicated links or as an overlay over IP. Such deployment approach is likely to create an heterogeneous environment in terms of supported network architectures at the access networks, paving the way for a new set of possibilities regarding the mobility of MNs. For example, a mobile node can not only move between access networks supporting the same network architecture, but also between IP-based access networks and FI-based access networks and between different FI-based access networks as well. New mobility management solutions targeting inter-network architecture mobility scenarios are required to enable MNs to preserve reachability to resources being accessed, whenever the MN changes the network architecture it is attached to. The access to resources can then be initiated in a given network architecture and be restored after each handover as the MN moves across access networks supporting different network architectures. As a result, the third objective to be approached by this thesis can be formulated as:

- **Objective 2a:** Design inter-network architecture mobility management mechanisms, by integrating both interoperability and mobility concepts, that allow MNs to move across different network architectures without losing the reachability to resources being accessed before the handover procedure.

1.3 Contributions

The developed work associated to this thesis addressed two main topics regarding the Future Internet research. More specifically, it focuses on interoperability and mobility research topics.

Regarding the first research topic, in order to reflect the importance of new network architectures to be able to exchange information with the existing networking environment, this thesis places a strong emphasis in enabling the interoperability between different Future Internet network architectures. To achieve this, an interoperability framework was designed to interconnect existing and upcoming Future Internet network architectures, allowing communication endpoints (e.g., hosts, services and contents) deployed in different network architectures to communicate and exchange information with one another. This enables the provision of transparent interoperability procedures for both network entities and user devices, as well as for existing mechanisms, services and applications, allowing not only each network architecture to evolve independently of the others, but also fostering the incremental deployment of new Future Internet network architectures. This work directly resulted in the following publications:

Carlos Guimarães, José Quevedo, Rui Ferreira, Daniel Corujo, Rui L. Aguiar, *Exploring Interoperability Assessment for Future Internet Architectures Roll Out*, Journal of Network and Computer Applications, Vol. 136, pp. 38-56, June 2019

José Quevedo, Rui Ferreira, **Carlos Guimarães**, Rui L. Aguiar, Daniel Corujo, *Internet of Things discovery in interoperable Information Centric and IP networks*, Internet Technology Letters, Vol. 1, No. 1, 2018

Daniel Corujo, **Carlos Guimarães**, José Quevedo, Rui Ferreira, Rui L. Aguiar, *Information Centric Exchange Mechanisms for IoT Interoperable Deployment* book chapter in "User-Centric and Information-Centric Networking and Services Access Networks, Cloud and IoT Perspective" pp. 71-140, CRC Press, May 2019

The second research topic of this thesis was centered on mobility, one of the key requirements that need to be addressed by the network architectures that compose the Internet of the future. The dynamics of wireless environments create complex challenges to ensure an always best connected connection for MNs. As such, in this thesis, mobility management mechanisms were designed for Future Internet. In particular, this work addressed mobility in both evolutionary SDN-based and clean-slate ICN architectures, by leveraging context information from both the user equipment and network to optimize the handover procedure on each network architecture. Such contribution enables not only to detect better handover candidates but also to optimize the handover procedure itself. Preemptive strategies can then be followed where current flows of a given MN are (re)configured towards its future network attachment location before the handover procedure actually occurs, minimizing the interruption delay caused by the handover procedure. Much of this work was reflected into the European project FP7 OFELIA[1] and open-source software ODTONE[2]. It directly resulted in the following publications:

Flávio Silva, Daniel Corujo, **Carlos Guimarães**, João Pereira, Pedro Rosa, Sergio Takeo, Augusto Neto, Rui L. Aguiar, *Enabling Network Mobility by Using IEEE 802.21 Integrated with the Entity Title Architecture*, Proc. 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribudos IV Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF), Brasilia, Brasil, May 2013

Carlos Guimarães, Daniel Corujo, Rui L. Aguiar, Flávio Silva, Pedro Rosa, *Empowering Software Defined Wireless Networks Through Media Independent Handover Management*, Proc. 2013 Globecom, Atlanta, USA, Dec 2013

Carlos Guimarães, Daniel Corujo, Flávio Silva, Pedro Rosa, Augusto Neto, Rui L. Aguiar, *IEEE 802.21-enabled Entity Title Architecture for Handover*

[1]OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) - http://www.fp7-ofelia.eu/

[2]Open Dot Twenty ONE (ODTONE) - https://atnog.av.it.pt/odtone

Optimization, Proc. WCNC 2014 IEEE Wireless Communications and Networking Conference, Istanbul, Turkey, Apr 2014

Carlos Guimarães, Daniel Corujo, Rui L. Aguiar, *Enhancing OpenFlow with Media Independent Management Capabilities*, Proc. ICC 2014 IEEE International Conference on Communications, Sydney, Australia, Jun 2014

The final research topic of this thesis focused on the integration of both interoperability and mobility concepts in a network environment where different network architectures coexist simultaneously. This integration paved the way for a new set of possibilities where MNs are able to move across access networks supporting different networks architectures and across access networks supporting the same network architecture. It is expected that, even after moving to a different network architecture, the MN is able to maintain the access to the content, avoiding the risk of segmentation on the content provisioning to a single network architecture. As such, in this thesis, another mobility framework targeting the scenarios where the MN actually moves across access networks supporting different network architectures was designed and implemented. MNs (and, consequently, applications) supporting multiple network architectures can initiate the download of a content on a given network architecture and, after moving to another network architecture, continue the download on its current network architecture. This work directly resulted in the following publication:

Carlos Guimarães, José Quevedo, Rui Ferreira, Daniel Corujo, Rui L. Aguiar, *Content Retrieval while Moving Across IP and NDN Network Architectures*, Proc. 24th IEEE Symposium on Computers and Communications (ISCC 2019), Barcelona, Spain, Jul 2019

An important aspect to take into consideration while reading this thesis is that due to validation and evaluation purposes, a more practical approach was followed where the proposed frameworks and mechanisms were exemplified, adapted and implemented over specific use cases targeting the selected network architectures. Nonetheless, the overall proposals in this thesis are generic and agnostic to any network architecture and they can be adapted to support the specificities of other network architectures.

1.4 Structure

The structure of this thesis, after this introduction, progresses with a background and related work chapter where the main concepts related with this thesis are presented. The main work developed under the scope of this thesis is described in the two following chapters, where the most novel contributions regarding the interoperability and mobility research topics are presented. Finally, the main conclusions and possibilities for future work are presented in the conclusion chapter.

The following chapters of this thesis are introduced by a short foreword which goal is to summarize the motivation for the given chapter. Additionally, each chapter is finalized with the main conclusion remarks, allowing the reader to review the main achievements and conclusions. The structure of this thesis is as follows:

- **Chapter 2 (Key Concepts and Technologies)** introduces the concepts required to contextualize the reader on the work presented throughout this thesis. More specifically, this section details aspects related with (i) Future Internet research, where special attention is given to evolutionary SDN-based and clean-slate ICN proposals; (ii) interoperability; and (iii) mobility.
- **Chapter 3 (Binding the Future Internet Architectural Landscape)** studies how different Future Internet and the current IP network architectures could coexist and interoperate with one another, in an environment where each architecture targets specific use case scenarios. However, to avoid the creation of information silos where information is restricted to the network architecture where it is deployed in, information needs to be available to entities independently of their access network architecture. A framework aiming to solve these challenges is proposed in this section. This framework is first evaluated in a simulation environment with results showcasing not only its practicability but also some faced limitations. This work progresses with the implementation of a proof-of-concept prototype deployed and evaluated under three use case scenarios: (i) web browsing; (ii) live streaming; and (iii) on-demand video delivery.
- **Chapter 4 (Supporting Mobility in Future Internet Architectures)** focuses on the study of mobility solutions for Future Internet architectures, both evolutionary and clean-slate. With mobility having an increasingly relevance in today's communications, it becomes crucial for new network architectures to fully support such requirement in order to provide users with an always best connected experience. As such, in this chapter, two mobility frameworks are defined, integrating Media Independent Handover mechanisms proposed by the IEEE 802.21 standard, for both evolutionary SDN-based and clean-slate ICN network architectures. In doing so, context from the link-layer of both MN and network entities can be used to optimize handover procedures in both network architectures, while providing a set of supporting mechanisms to control the different steps of the handover procedure. Controlling entities can

then dynamically and preemptively reconfigure the network to cope with the changing location of MNs, minimizing the impact to ongoing data sessions and increasing connectivity opportunities. This chapter progresses with the study of mobility scenarios where MNs are able to move across access networks supporting different network architectures (e.g., IP and ICN). The edge of the network is an expected location for the initial roll out of native deployments of new network architectures (including clean-slate network architectures), which would create an heterogeneous Future Internet environment in terms of supported network architectures at the access networks. As such, an inter-network architecture mobility framework is also designed, implemented and validated, allowing MNs to move across IP and ICN network architectures in a content retrieval use case scenario without applications losing access to contents.

- **Chapter 5 (Conclusions and Future Work)** overviews the main conclusions and achievements resulting from the developed work. It also expands beyond the reach of this thesis, by providing insights on the upcoming challenges which may directly or indirectly benefit from the presented work.

Chapter 2

Key Concepts and Technologies

While the previous chapter presented the motivation, goals and outcomes of this thesis, this chapter aims at contextualizing the reader towards the work developed in this thesis. It presents an overview on Future Internet research, focusing on deployment and interoperability as well as on mobility aspects. In doing so, its purpose is to pave the way to the thesis work, providing the building blocks of the developed work.

2.1 Introduction

Despite that both interoperability and mobility have been addressed in the past for the IP architecture, the proposal of new network architectures for the Future Internet architecture placed both topics back into the attention of the research community. The fundamental problems on each subject are still the same independently of the target network architecture(s) and, therefore, understanding the state of the art is a critical aspect before considering the design of new solutions for the Future Internet architectures. As such, this chapter presents the relevant base concepts regarding Future Internet research, interoperability and mobility, along with the relevant state of the art, in context with the work developed within this thesis.

2.2 Future Internet Research

Research on Future Internet has been gaining traction on the recent years, motivated by the changing requirements of the current Internet. The Internet was not originally designed to cope with e.g. the wide variety and number of existing devices and applications, the capability of such devices to be mobile, the ever growing amount of exchanging traffic in the Internet and the need for intrinsic security mechanisms. The limitations of the current Internet are increasingly being recognized within the research community, with the need for change being acknowledged [82]. However, how such change should take place is still under discussion, with several solutions being proposed by the research community.

Future Internet research has been following two main trends for the design of its architecture [93, 94]. On one hand, evolutionary approaches focus on the understanding and evolution of the current existing Internet architecture as the approach to design the Future Internet architecture. On the other hand, clean-slate approaches aim on the design of new network architectures from scratch without any bounds to the current existing Internet architecture.

A critical aspect for the adoption of the designed Future Internet architectures is their assessment and evaluation at scale and under realistic conditions. For that, large-scale experimental testbeds have been proposed (Global Environment for Network Innovations (GENI) [47] and Future Internet Research and Experimentation (FIRE) [46] are two of the biggest efforts) so that proposed Future Internet architectures can be tested, validated and improved, before being deployed in the real world [84].

2.2.1 Software Defined Networking

Software Defined Networking (SDN) [68] is an emerging evolutionary architectural paradigm for the Future Internet, which presents a different way of operating the network infrastructure. Its operation is based on the separation of the control plane from the data plane, deploying network nodes capable of recognizing and applying intelligent behavior to traversing packets. Network forwarding equipment (e.g. switches and routers), such as the SDN switch, becomes a simple forwarding device, with the control decisions implemented in a logically centralized controller, called the SDN controller. In other words, the system that makes decisions about where the traffic is sent is disassociated from the SDN switches, and is instead deployed in the SDN controller. Such separation not only brings several benefits compared with legacy methods (e.g., makes the networks more agile and flexible), but also introduces new possibilities regarding network management and configuration (e.g., allows network administrators to manage the network in a centralized way, without step in the configuration of each individual switch).

By moving the control procedures to a logically centralized entity, SDN can exploit the benefits of centralized solutions. Network operators do not need to individually configure each network equipment, but instead make network-wide traffic forwarding decisions in the SDN controller [63], which are then automated by the SDN controller and further implemented in the SDN switches. The SDN controller, by having a global view of the network state under its domain, can make better decisions regarding network (re)configuration, adding a greater degree of flexibility to the operation of the underlying network, allowing the network to better react to dynamic environments by (re)configuring it on-the-fly and according to the demands on each instant.

In addition, SDN can facilitate innovation at the network level. By having the control plane running in software and separated from the forwarding devices, new e.g. protocols and algorithms can be implemented in software without requiring modifications in the network equipments.

To achieve its purpose, well-defined application programming interfaces (APIs) are defined, namely the southbound and the northbound interfaces, as depicted in Figure 2.1. These APIs allow the SDN controller to communicate with the data plane and the application plane, respectively. More specifically, the southbound API allows the SDN controller to communicate with the SDN switches, allowing it to manage the underlying data plane by directly controlling the state of different elements of an SDN switch (such as, the forwarding tables). In turn, the northbound interface enables the communication between network applications and the SDN controller, determining how operational tasks and network policies are expressed and providing network applications an abstraction of the underlying physical network.

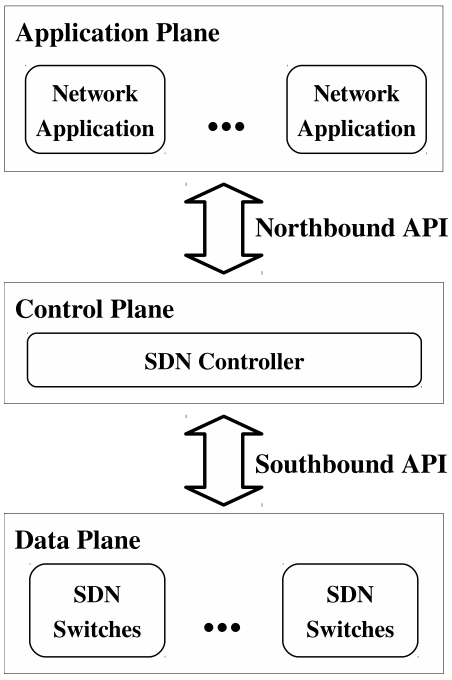


Figure 2.1: Overview of SDN architecture

2.2.1.1 Southbound API

OpenFlow [75] is a protocol to support the SDN paradigm, being one of the most popular protocol standards of the SDN southbound API, with several evolving versions of its specification being standardized over the recent years. Originally designed for network researchers to test their ideas in real production networks, OpenFlow succeeded in attracting commercial vendors, being already available in a number of commercial products [70]. It promotes the separation of the networking intelligence from the forwarding devices, to be deployed in a logically centralized controller. As such, in an OpenFlow network, a single controller can control multiple switches, which allows it to have an holistic view of the underlying network.

The OpenFlow architecture is composed by three main components: (i) the OpenFlow controller; (ii) the OpenFlow switch; and (iii) the OpenFlow channel.

The OpenFlow controller is the central entity of an OpenFlow network responsible for configuring the behavior of the forwarding plane of OpenFlow switches. For that, it not only has a global view of the network topology under its domain, but it can also access statistics about the flows crossing the OpenFlow switches. Using this knowledge, the OpenFlow controller can dynamically (re)configure the network by updating the flow tables of the OpenFlow switches.

The OpenFlow switch is responsible for executing data packet forwarding operations. To achieve its purpose, it is composed by flow tables which are responsible for storing the flow entries, each having information on how to process incoming packets. As such, flow entries correspond to rules that are configured in the switch and that specify how flows are handled and processed. In OpenFlow, the definition of a flow goes beyond the typical IP source/destination tuple. Flows are also identified by other fields belonging to the packet (e.g., Ethernet source and destination addresses, Ethernet frame type and IPv4/IPv6 source and destination addresses). Each flow entry is composed by three elements: match fields, counters and instructions. The match fields are used to match incoming packets to the correspondent flow entry. The counters (e.g., number of packets and bytes) are related with statistics about each flow entry. The instructions consist on a list of actions (e.g., modify/update packet headers, specify the output interface and send packet to controller) to be applied to incoming packets matching the corresponding flow entry.

Finally, the OpenFlow channel enables the communication between the OpenFlow controller and OpenFlow switches via the OpenFlow protocol. The messages from the OpenFlow protocol are divided into three main categories: (i) *Controller-to-Switch*; (ii) *Asynchronous*; and (iii) *Symmetric*. The *Controller-to-Switch* messages are initiated by the OpenFlow Controller and may or may not have a response back from the OpenFlow Switch. The *Asynchronous* messages are proactively sent from the OpenFlow Switches without any request from the OpenFlow Controller, to notify about packet arrival or changes on the state of the OpenFlow Switch. Finally, the *Symmetric* messages are sent by either the OpenFlow Switch or the OpenFlow Controller without being requested by neither one.

The configuration of rules in the OpenFlow switches by the OpenFlow controller can follow two different approaches [81, 38]: (i) reactive; or (ii) proactive.

In the reactive approach, rules are configured in the OpenFlow switches on demand. After the reception of a packet belonging to a new flow (i.e., without any match on its flow tables), the OpenFlow Switch encapsulates the received packet in an *OFPT_PACKET_IN* message to be sent to the OpenFlow Controller. Such message triggers the OpenFlow Controller to compute the rule associated to the new flow, installing it in the OpenFlow Switch via *OFPT_FLOW_MOD* message. Subsequent packets of the same flow arriving the OpenFlow Switch will be processed internally according to the rules installed in its flow tables, without requiring the intervention of the OpenFlow Controller. Alternatively, the OpenFlow Controller, instead of configuring the rules in the OpenFlow switch after the reception of the *OFPT_PACKET_IN* message, can opt to send a packet (which can be the received packet in the *OFPT_PACKET_IN* message or a completely new packet) out through the datapath. For that, it encapsulates the packet to be sent in an *OFPT_PACKET_OUT* message, which by being received by the OpenFlow Switch will be desencapsulated and forwarded towards the datapath. However, reactive approaches introduce additional latency to incoming packets, resulting from the communication with the OpenFlow Controller.

Alternatively, in the proactive approach, forwarding rules are preemptively installed in the OpenFlow Switch by the OpenFlow Controller, before the first packet of corresponding flows arrives the OpenFlow Switch. This strategy is commonly used for accessing control procedures, where rules are known *a priori*. For populating the flow tables of the OpenFlow switches, the OpenFlow Controller issues *OFPT_FLOW_MOD* messages towards the underlying OpenFlow Switches with information regarding the flow matching and actions to be applied to transversing packets. Such approach not only avoids traffic disruption in case an OpenFlow switch loses its connection with the OpenFlow controller, but also avoids the latency introduced by flow setup procedures. However, it requires predicting in advance which network traffic will eventually transverse the OpenFlow network so that it could be configured accordingly.

In Figure 2.2, a configuration procedure of an OpenFlow network using both reactive and proactive approaches is presented.

Whenever a data packet is received by an OpenFlow switch on the edge of an OpenFlow domain (message 1 in Figure 2.2), if a match for the corresponding flow is not found on its flow tables, the OpenFlow Switch triggers the reactive procedures for inquiring the OpenFlow Controller about the forwarding rules corresponding to that flow. It encapsulates the received packet in an *OFPT_PACKET_IN* message to be

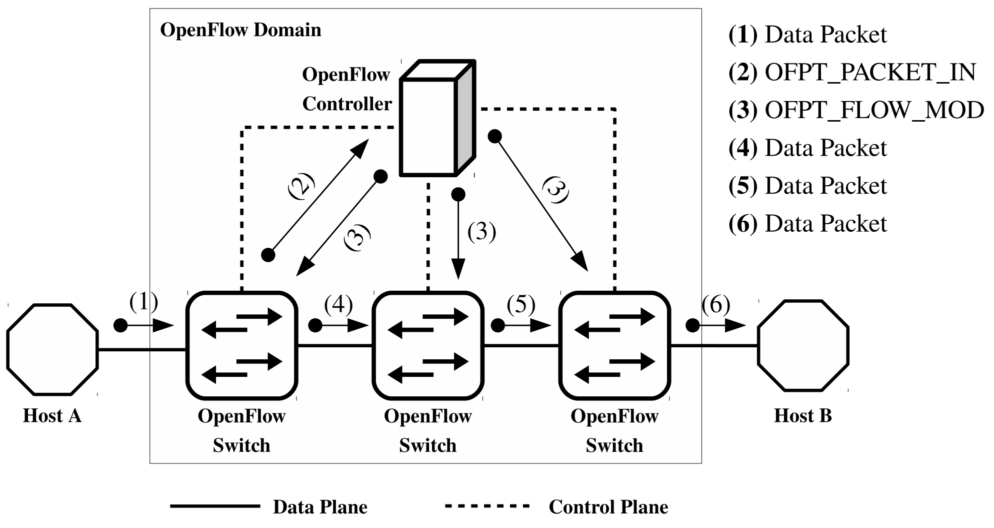


Figure 2.2: Example of a configuration procedure of an OpenFlow network

sent to the OpenFlow Controller (message 2 in Figure 2.2). The OpenFlow Controller computes the forwarding rules not only for the requesting switch but also for the remaining OpenFlow Switches through which the packet will transverse until it exits the OpenFlow domain. These rules are then installed in the OpenFlow Switches using *OFPT_FLOW_MOD* messages (messages 3 in Figure 2.2). The configuration of the first OpenFlow Switch follows an reactive approach, while the remaining OpenFlow switches can then be configured following a proactive approach. In doing so, the additional delay resulting from the setup procedures occurs only once, instead of for each hop in the path inside the OpenFlow domain. After the configuration of the OpenFlow Switches, the data packet is forwarded towards its destination (messages 4, 5 and 6 in Figure 2.2).

2.2.1.2 Northbound API

The Northbound API is an interface provided by the SDN Controllers to the network application developers, with the purpose of abstracting the low level instruction set used to configure forwarding state of the SDN Switches via Southbound API, while allowing concurrent access to the underlying network and enforcing security mechanisms. In opposite to the Southbound API where OpenFlow emerges as the *de facto* standard, in the Northbound API there is not a standard. Instead, each SDN Controller provides a well-defined interface (i.e., the Northbound API) to network application developers via REST APIs, ad-hoc APIs or other types of APIs.

This API commonly provide a set of generic functionalities to the network application developers, such as network topology and network state information, device discovery and abstraction of the network configuration. Since none of the available interfaces as yet arised as a dominant choice for the Northbound API, instead of providing a detailed analysis of each one, a list of the most popular SDN Controllers is enumerated: OpenDayLight[1], NOX[2][54], POX[3], Floodlight[4], Ryu[5], Trema[6], among others [68].

2.2.2 Information-Centric Networking

Information-Centric Networking (ICN) [110, 10] is an emerging clean-slate architectural paradigm for the Future Internet, which has been acquiring a significant relevance in the research community by both the academia and industry. In ICN, users are interested in what the content is instead of where it comes from [112]. As such, ICN places the content itself as the central networking element of its architecture, moving away from the host-oriented and end-to-end principles of the current Internet. This change is mainly motivated by the increasing interest in consuming information independently of its location, rather than in communicating with specific hosts.

An advantage of the ICN paradigm is that it leverages native support for scalable and efficient content dissemination. It locates and routes content based on its name, decoupling it from its location. In doing so, ICN focuses in securing the content, making it trustworthy by itself, instead of securing the path as in the current host-centric approach. The validity of content can be verified directly, and not by checking which host provided the content.

By leveraging these two properties, ICN enables in-network caching. Any entity in the network can maintain a copy of contents, acting as a cache. Requests for a given content can be satisfied not only by the original information source, but also by any entity holding a cached copy of the content in the path between the requester and original information source. Since ICN focuses on securing the content itself instead of the communication channel, it can be retrieved by untrustworthy entities and through untrustworthy connections.

ICN also leverages multicast communications for retrieving content, since by naming information at the network-layer, it allows the aggregation of requests for the same content. Finally, since communication in ICN is mostly receiver-driven, mobility appears as an intrinsic functionality. When a consumer moves to a new location, it re-issues requests that have not been satisfied yet. However, mobility of the content

[1]OpenDayLight - https://www.opendaylight.org/
[2]NOX - https://github.com/noxrepo/nox
[3]POX - https://github.com/noxrepo/pox
[4]Floodlight - http://www.projectfloodlight.org/
[5]Ryu - https://osrg.github.io/ryu/
[6]Trema - https://trema.github.io/trema/

source is not so straight-forward, imposing a new set of challenges [101], due to the change of the location of contents.

Over the years, several instantiations of the ICN paradigm have been proposed, such as Named Data Networking (NDN) [113], Content Centric Networking (CCN) [59], Network of Information (NETINF) [32], Publish Subscribe Internet Technology (PURSUIT) [98] and Entity Title Architecture (ETArch) [33]. Although sharing the same core functionalities of this novel paradigm, each instantiation differ from the others on how such functionalities are implemented.

Despite the benefits that the ICN paradigm can bring, its deployment requires a larger investment and agreements among different stakeholders, when compared with incremental solutions over the current Internet architecture. Its success will then depend on its capability to provide clear superior solutions to well-known problems in the current IP network architecture, as well as its capability to be incrementally deployed along with the current Internet architecture.

The interest of the research community in ICN led to the creation of a Information-Centric Networking Research Group (ICNRG)[7] discussion group by the Internet Research Task Force (IRTF).

In the following subsections, the operation of the NDN, PURSUIT and ETArch instantiations are detailed, due to its relevance on the work performed in this thesis.

2.2.2.1 Named Data Networking

Named Data Networking (NDN) [113] is an instantiation of the ICN paradigm that can be categorized as an Interest-based ICN solution. Interest-based ICNs propose a communication model driven by consumers, operating in a request/response communication paradigm, where information is polled using two different messages: (i) *Interest* messages, which are used to request desired content; and (ii) *Data* messages, which are used to carry the data itself.

Instead of carrying source and destination addresses as in the current IP architecture, both the *Interest* and *Data* messages contain a name used to identify the content being addressed. The names in NDN are hierarchically structured and human-readable (e.g., "*/atnog.av.it.pt/content/video.mpg*"), although in special cases flat names can also be accommodated in the NDN design. Also, NDN introduces the notion of a "face", which consists on a generalization of a network interface and may represent a physical network interface, an overlay communication channel or an inter-process communication channel with the applications.

[7]Information-Centric Networking Research Group (ICNRG) - https://irtf.org/icnrg

An NDN-enabled node is composed by three main modules: (i) Content Store (CS), which consists on a cache for contents; (ii) Pending Interest Table (PIT), which maintains track of yet unsatisfied *Interest* packets; (iii) Forwarding Information Base (FIB), which contains a set of forwarding rules for *Interest* packets. The basic operation of NDN is depicted in Figure 2.3.

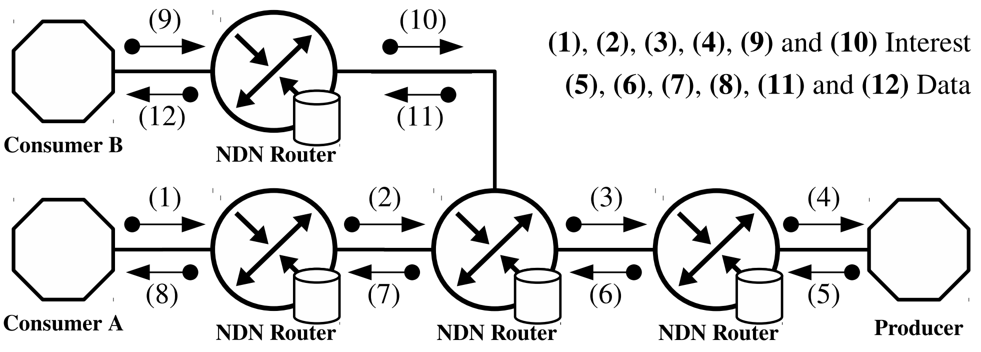


Figure 2.3: Example of the basic operation of NDN

Consumers issue *Interest* messages addressing the desired content through its name (message 1 in Figure 2.3). Upon reception by an NDN router node, the node verifies if it contains a cached copy of the content in the CS. If a cached copy exists, the node itself can reply to the *Interest* messages with a *Data* message, even if it is not the producer of the content. Otherwise, if the content is not cached in the CS, the node checks if a pending request for the same content exists in the PIT. If so, the incoming face of the *Interest* message is added to the PIT entry and the *Interest* message discarded. Otherwise, if no entry exists in the PIT, the node creates the PIT entry (i.e., association between the content name and incoming face of the *Interest* message) and forwards the *Interest* message based on the information contained in the FIB (messages 2 to 4 in Figure 2.3).

The *Data* message follows the reverse path taken by the corresponding *Interest* message (messages 5 to 8 in Figure 2.3). Upon reception of the *Data* message, an NDN router node lookups on its PIT to obtain the faces from which it received the *Interest* messages for that content. If no entry is found in the PIT, the *Data* message is discarded. Otherwise, the *Data* message is sent according to the state information stored in the PIT entry (i.e., via the faces through which it received the *Interest* messages). The *Interests* messages are considered as satisfied (i.e., *Data* consumes *Interests*), and the corresponding PIT entry removed. According to the caching policies configured in the node, content objects can be cached in the CS.

In the example presented in Figure 2.3, a second consumer (i.e., consumer B) issues an *Interest* message addressing the same content (messages 9 and 10 in Figure 2.3).

In this case, the reply with the desired content is sent by an NDN router in the path between the consumer and the producer (messages 11 and 12 in Figure 2.3), since it has previously cached the content during the request for the same content by consumer A.

2.2.2.2 Publish Subscribe Internet Technology

Publish Subscribe Internet Technology (PURSUIT) [98], a follow-up of the Publish-Subscribe Internet Routing Paradigm (PSIRP) project [43, 42], presents an ICN architecture on which publish/subscribe primitives play a central role in its operation. Instead of establishing connections between specific nodes as in the current IP architecture, in PURSUIT information is published and interested consumers subscribe to it.

Content is named in PURSUIT by a unique pair of identifiers, namely the scope and the rendezvous identifiers. The scope may represent a physical or a logical structure, grouping related information objects, and consists on a (sequence of) fixed length *scope ID(s)*, while the rendezvous identifier uniquely identifies the content within the respective scope, consisting of a single fixed-length *rendezvous ID*.

The basic operation of PURSUIT is depicted in Figure 2.4, where a consumer requests access to a given content object previously published by its producer.

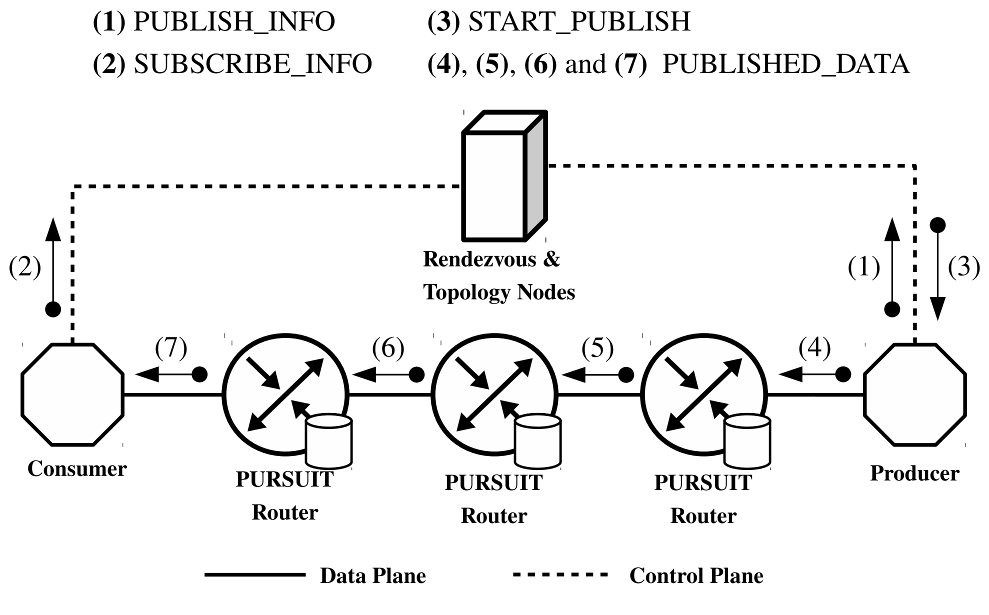


Figure 2.4: Example of the basic operation of PURSUIT

Content producers (also known as publishers) advertise content availability towards the *Rendezvous Node* using a *PUBLISH_INFO* message (message 1 in Figure 2.4). Consumers (also known as subscribers) interested in a given content send a *SUBSCRIBE_INFO* message towards the *Rendezvous Node* (message 2 in Figure 2.4).

The *Rendezvous Node* is responsible for matching publication and subscriptions for a given content, requesting the *Topology Node* to calculate a forwarding identifier corresponding to a path between the publisher and the subscriber(s). The forwarding identifier consists on a LIPSIN identifier [61]. This identifier is sent to the publisher via a *START_PUBLISH* message (message 3 in Figure 2.4) which triggers the publisher to send the content to interested subscriber(s) via *PUBLISHED_DATA* message (messages 4 to 7 in Figure 2.4).

Additionally, the extensions proposed in [96, 97] allow the subscriber and the publisher to communicate in a request/response approach using the publish/subscribe primitives. The operation of PURSUIT using these extensions is depicted in Figure 2.5.

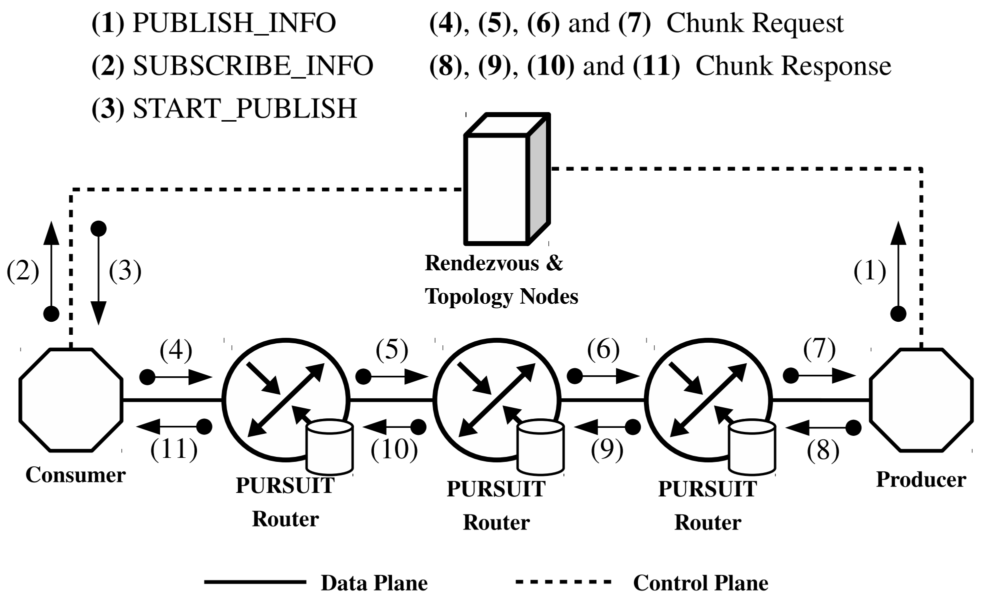


Figure 2.5: Example of the basic operation of PURSUIT extensions proposed in [96, 97]

In this case, whenever the consumer subscribes (message 2 in Figure 2.5) its interest on a previously published content (message 1 in Figure 2.5), the *Topology Manager* calculates both the forwarding and reverse direction identifiers between the subscriber and the publisher. Both forwarding identifiers are then delivered to the subscriber (message 3 in Figure 2.5). The subscriber uses the reverse direction identifier to forward *Chunk Requests* messages encapsulated in *PUBLISHED_DATA* messages towards the publisher (messages 4 to 7 in Figure 2.5). The forwarding direction identifier, that can be used by the publisher to reply to the request using a *Chunk Response* message, is included in the message encapsulated in a *PUBLISHED_DATA* message, sent directly to the subscriber (messages 8 to 11 in Figure 2.5).

2.2.2.3 Entity Title Architecture

The Entity Title Architecture (ETArch) [33] is a clean-slate network architecture that shares the vision of the information-oriented paradigm, where users request information by attaching to the correspondent *Workspace*, triggering the network to dynamically configure itself to provide users with the desired content.

ETArch employs a new naming and addressing scheme based on a topology-independent designation, referred as *Title*. A *Title* is used to uniquely identify an *Entity*, but an *Entity* can still be identified by multiple *Titles*. An *Entity* has communication requirements and capabilities and it can assume multiple representations, including contents, services, applications, hosts, among others. For *Entities* to be able to exchange information with one another, they have to be attached to the same communication channel, called *Workspace*. *Workspaces* represent the path where data is transported to all entities attached to such communication channel and, therefore, they gather multiple communication entities, so that whenever a message is sent to a given *Workspace* it will be received by every entity attached to that *Workspace*.

A key component of this architecture is the *Domain Title Service* (DTS), which consists on a distributed system that deals with all control aspects of the network. The DTS is composed by *Domain Title Service Agents* (DTSAs) responsible for maintaining information about *Entities* registered in the domain and the *Workspaces* that they are subscribed to, with the purpose of configuring the underlying network devices to implement the *Workspaces*, allowing data to reach every subscribed entity. As such, whenever a new *Entity* attaches to a given *Workspace*, the DTSA configures the forwarding tables of the underlying equipment with the purpose of extending the *Workspace* towards the new *Entity*. Likewise, whenever an *Entity* detaches from a given *Workspace*, the DTSA is responsible for removing the forwarding rules from the underlying equipment. For allowing the communication between the DTSA and the communication endpoints for registering and attachment procedures, the ETArch defines the Entity Title Control Protocol (ETCP).

The operation of ETArch, on which a centralized entity configures the behavior of the forwarding plane, relies on SDN concepts, which are implemented in ETArch by the OpenFlow protocol. DTSAs act as OpenFlow controllers capable of (re)configuring the forwarding tables of the underlying network (i.e., OpenFlow switches) in order to implement the *Workspaces*. In other words, the DTSA is responsible for configuring the path between all *Entities* attached to the same *Workspace*. The basic operation of ETArch is depicted in Figure 2.6.

A producer of new content willing to provide it to other entities starts by registering itself in the DTSA (message 1 in Figure 2.6), after which requests the creation

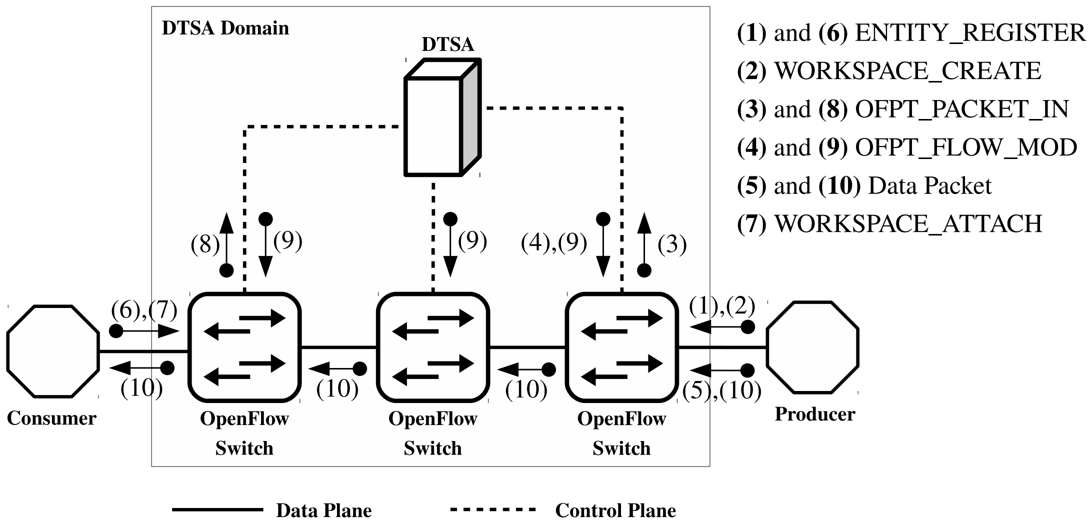


Figure 2.6: Example of the basic operation of ETArch

of an *Workspace* by issuing an *WORKSPACE_CREATE* message (message 2 in Figure 2.6). The OpenFlow switch, by receiving this request, encapsulates it in an *OFPT_PACKET_IN* message to be sent to the DTSA (message 3 in Figure 2.6). The DTSA creates the corresponding *Workspace*, implementing it in the underlying network using *OFPT_FLOW_MOD* messages (message 4 in Figure 2.6). At this point, the producer can start publishing contents to the *Workspace* but, since no other entities are attached to the *Workspace*, messages are not propagated further than the first OpenFlow switch (message 5 in Figure 2.6). In turn, a consumer desiring to access the contents published by the producer starts by registering itself in the DTSA (message 6 in Figure 2.6), after which requests the attachment to an *Workspace* by issuing an *WORKSPACE_ATTACH* message (message 7 in Figure 2.6). As for the creation of the *Workspace*, this message is encapsulated in an *OFPT_PACKET_IN* message by the OpenFlow switch and sent to the DTSA (message 8 in Figure 2.6). The DTSA computes and installs in underlying OpenFlow switches the forwarding rules required to extend the *Workspace* towards the consumer (message 9 in Figure 2.6). After configuring the extension of the *Workspace*, new messages published by the producer are forwarded towards the consumers (message 10 in Figure 2.6).

2.2.2.4 Brief comparison of IP and selected ICN architectures

The main differences between IP, NDN, PURSUIT and ETArch architectures, which have the potential to hinder their interoperation, are presented in Table 2.1.

While the IP network architecture is host-oriented and thus network layer addresses represent the endpoint holding the resources to be accessed, NDN, PURSUIT and ETArch are based on a data-oriented paradigm, enabling addresses to represent

Table 2.1: Comparison between IP, NDN, PURSUIT and ETArch architectures

	Design Orientation	**Communication Paradigm**	**Network Addresses**	**Network Protocol**
IP	Host-oriented	Push	(i) Fixed length (ii) Hierarchical identifier	Single *IP* message
NDN	Data-oriented	Pull (Request/Response)	(i) Variable length (ii) Hierarchical identifier (iii) (Optional) Human-readable identifier	*Interest* and *Data* messages
PURSUIT	Data-oriented	Pull (Publish/Subscribe)	(i) Variable length: composed by a (sequence of) scope(s) ID(s) and a single rendezvous IDs (ii) Flat identifier	*(Un)Subscribe* and *Publish* messages
ETArch	Data-oriented	Pull (Publish/Subscribe)	(i) Fixed length (ii) Flat identifier	*Entity (un)registration* and *Workspace create/delete*, *attach/detach* and *publish* messages

anything (e.g., a data chunk, a content or even an endpoint). The IP addresses are hierarchical and encoded as a sequence of 4 bytes for IPv4 or 16 bytes for IPv6. Although NDN addresses are also hierarchical, they can have variable lengths. In turn, PURSUIT addresses are flat and composed by two parts: the scope and the rendezvous. The scope part, consisting of a (sequence of) fixed length *scope ID(s)*, is followed by the rendezvous part, consisting of a single fixed-length *rendezvous ID*. Finally, ETArch addresses have fixed length and consists in flat identifiers.

In terms of operation, the IP protocol is composed by a single message that can be sent and routed in the network independently of any other message previously sent. The NDN protocol, operating on a request/response paradigm, is composed by two types of messages which are intrinsically related (i.e., *Data* messages are forwarded through the reverse path taken by *Interest* message). The PURSUIT protocol, which operates based on a publish/subscribe paradigm, has specific messages for handling the (un)subscriptions and publication of data, forwarded based on in-packet bloom filters. ETArch, that also operates based on a publish/subscribe paradigm, defines messages for entity (un)registration as well as for enabling entities to create or attach to workspaces (i.e., subscription) in order to become capable of receiving any message sent towards a given workspace (i.e., publish).

2.2.3 Discussion

Future Internet is a recent research topic which is currently having a strong emphasis among the research community, with several network architectures and enhancements to those same architectures being proposed over the recent years. Two different strategies for the definition of the Future Internet are being taken. On one hand, evolutionary approaches (such as, SDN-based solutions) aim to enhance the current Internet architecture, maintaining backwards compatibility with the existing Internet infrastructure. On the other hand, clean-slate approaches (such as, ICNs) aim to design new network architectures from scratch without any bounds to the current existing Internet architecture. Each of these architectures optimizes different KPIs by centering their operation on different network elements (e.g., IP is centered on hosts, while ICNs center their operation around the content itself) and/or by defining different ways for establishing the communication between the endpoints. Nevertheless, since such network architectures are still in the maturing phase, there is space for new innovations that could be incorporated in their base design.

2.3 Deployment and interoperability

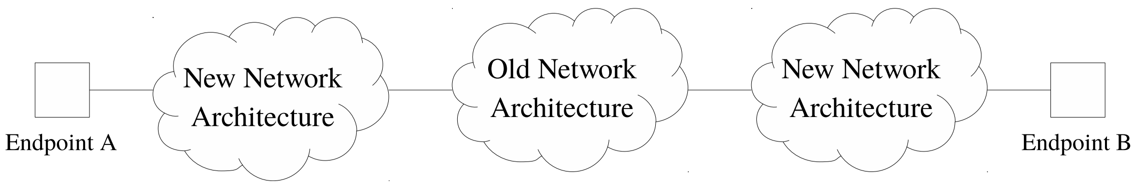
The IP protocol prevailed as the main protocol at the network layer, having gradually imposed itself on all types of networks. In doing so, IP became the narrow waist of the current Internet, leading to a new concept based on "all-IP" networks [9]. The layers above and below the network layer have seen frequent innovations, with new protocols being proposed on those layers. In contrast, such evolution pattern is not witnessed at the network layer, where the adoption of new protocols is a costly and slowly process.

2.3.1 Deployment scenarios

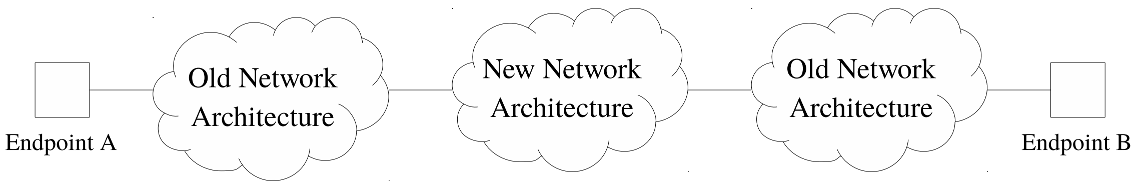
During the transition from one network architecture to another, it is expected a transition period on which both the old and the new network architectures would run in parallel. In an ideal deployment of new network architectures, both the new and existing architectures would be supported in all network equipments (i.e., following a dual stack strategy), with each network architecture being natively used whenever needed. However, such a strategy may increase not only the costs of replacing/upgrading existing equipments to support the new network architecture but also the operational costs due to duplication of tasks for maintaining both the new and the old network architecture.

Notwithstanding, during the deployment of a new network architecture, three deployment scenarios can be envisioned, as depicted in Figure 2.7. Note that this is well known from the issues associated with the IPv6 transition.

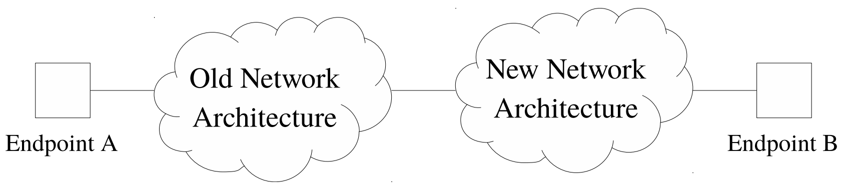
In early deployment stages, isolated networking islands supporting the new architecture would be incrementally deployed over time. While the new network



(a) Interconnecting new network architecture islands through the old network architecture



(b) Interconnecting old network architecture islands through the new network architecture



(c) Interconnecting the new and the old network architectures

Figure 2.7: Deployment scenarios

architecture is being deployed, the existing networking infrastructure can remain functional and be used to carry traffic of the new network architecture. As such, besides interconnecting islands through dedicated links, the interconnection of these new islands can be achieved by transporting traffic between islands over the existing networking architecture, as depicted in Figure 2.7a. For example, the IPv4 routing infrastructure remains functional while the IPv6 infrastructure is being deployed, with the former being used to carry traffic from the latter to interconnect isolated islands [48].

In later stages of deployment of a new architecture and with the new architecture widely adopted, it is expected that the old network architecture becomes a legacy service, starting to be no longer supported in parts of the network. With time, the old network architecture may become scattered among multiple isolated islands, connected to each other via the new network architecture, as depicted in Figure 2.7b.

A third deployment scenario, that can occur in every stage of the deployment of a new network architecture, considers that a given endpoint from the new network architecture needs to exchange information with an endpoint from the old network architecture (and vice-versa), as depicted in Figure 2.7c. In this scenario, messages from one network architecture need to be converted into compatible messages with the other network architecture.

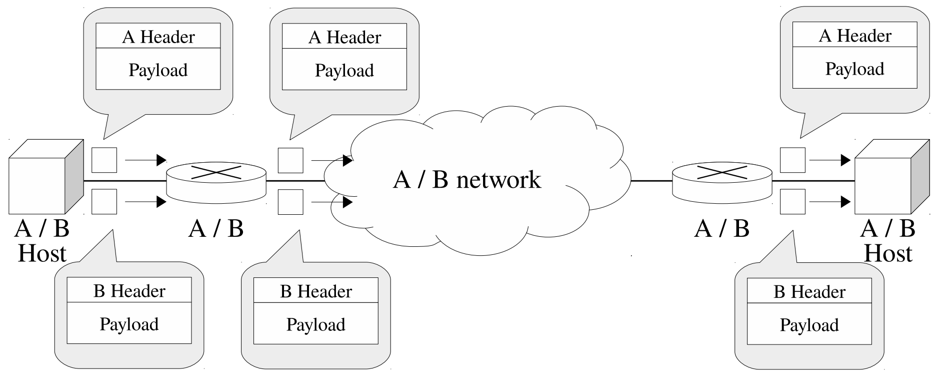
2.3.2 Interoperability strategies

In computer networks, the *flag day* is a transition strategy on which all existing network entities are simultaneously upgraded to support a new feature or protocol. It involves a period of outage, where network entities often need to be rebooted. Those that are not upgraded are left having to communicate via ad-hoc mechanisms. In the history of the Internet, such event occurred seldom e.g. in 1983 when ARPANET switched from the Network Control Program (NCP) to the TCP/IP protocol suite [71]. Back then, the number of hosts were just a few hundreds. Even so, there was a transition period between both protocols before such change occured [86], where it was considered that NCP-only hosts needed to interoperate with IP/TCP-only hosts. The solution relied on the introduction of a set of relaying hosts supporting both NCP and IP/TCP protocols, that intermediated the communication between hosts for Telnet, FTP and Mail protocols.

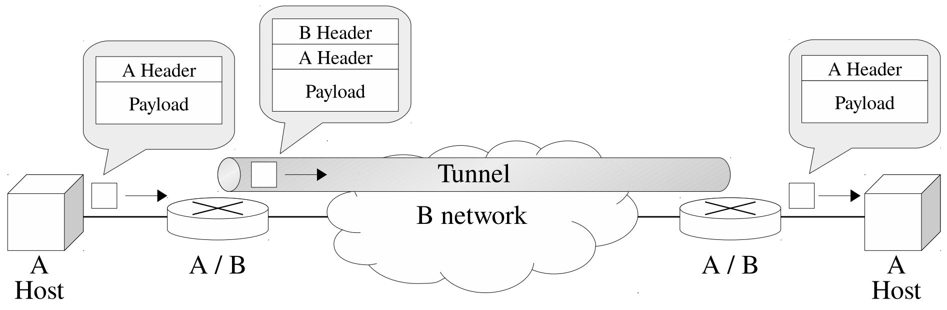
Today, it is practically impossible to switch the entire Internet to use a new network architecture overnight. Not only the number of existing network entities is far superior than in 1983 (from a couple hundreds to hundreds of millions machines) but also it would require the agreement of different stakeholders. Interoperability mechanisms between different network architecture are then required, so that the different protocol stacks of each architecture can coexist with one another while enabling an incremental deployment of new architectures in a transparent way for end users. During such transitional time, it is expected that both the existing networking landscape and the new network protocol stack coexist and operate in parallel. To achieve such purpose, three different interoperability strategies have been employed while deploying a new network architecture (e.g., during the transition time from IPv4 to IPv6 [108, 103]), as depicted in Figure 2.8: (i) dual-stack; (ii) tunneling; and (iii) translation strategies.

The most straightforward solution is to follow a dual-stack strategy where both the new and the existing protocol stacks are simultaneously supported by both the hosts and routers (Figure 2.8a). This implies that a complete implementation of each protocol stack is provided by the network entities, which by operating in parallel, allows networks entities to communicate via either protocol. Such solution paves the way for services and applications supporting the new protocol to start to emerge. In the example of Figure 2.8a, all entities support both network protocol A and B and, therefore, endpoints can use either protocol A or B to send messages towards the other communication endpoint. However, such solution has the disadvantage of creating a complexity not only in what concerns the design of network entities but also due to a double effort in configuring and maintaining of multiple network protocol stacks.

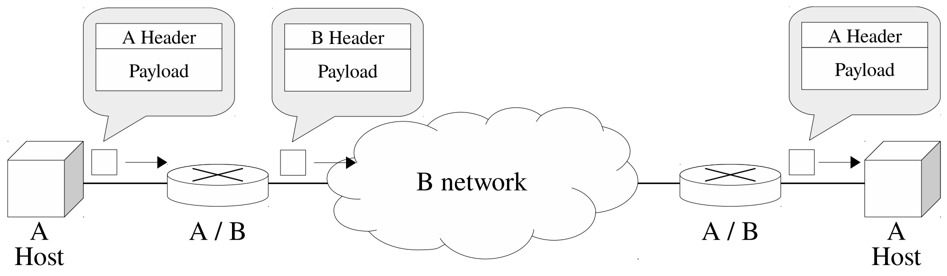
It is expected that during the transition period, isolated islands of the new network



(a) Dual Stack



(b) Tunneling



(c) Translation

Figure 2.8: Interoperability strategies

architecture are going to need to use the existing networking infrastructure to connect to each other. In such scenarios, islands of a given network architecture can be connected through tunnels that route packets on top of the existing networking environment, without requiring its upgrade (Figure 2.8b), creating virtual links between the islands. At the endpoint of each tunnel are entities able to understand both network protocols. Packets from the new network architecture are encapsulated in the entrance point of the tunnel and decapsulated at the tunnel exit point. As such, tunneling provides a way to utilize the existing routing infrastructure to carry packets of non-supported architectures. In the example of Figure 2.8b, the gateway, supporting both A and B protocols, adds a header of the protocol B to received packets of protocol A, so that they can be forwarded towards the other tunnel endpoint via routing infrastructure of architecture B. In turn, in the tunnel exit point, which also supports both A and B

protocols, the header of the protocol B is removed from the received packets, forwarding the original packet in the network architecture A towards its original destination.

Another strategy to connect isolated islands of a given network architecture is to translate packets from one network protocol to another (Figure 2.8c). This strategy also enables network entities belonging to a different network architecture to communicate with one another. Translation mechanisms can be stateless or stateful. In the former, each translation can be processed individually without any reference to previously converted packets, while the latter requires state regarding previous translations to be maintained. However, this strategy has the disadvantage that an unclear mapping between the features of each network protocol may result in feature loss. In the example of Figure 2.8b, the gateway, supporting both A and B protocols, extracts the payload of the message, converting the header of protocol A into an header of protocol B, so that the message can be forwarded across the infrastructure supporting network architecture B. The message is then converted back to an A message at the edge of network B and delivered to its original destination in the network architecture A. Moreover, this strategy allows messages to be delivered to an entity in the architecture B. In this case, the second conversion (i.e., from architecture B back to architecture A) does not occur.

2.3.3 Interoperability mechanisms between IPv4 and IPv6

Along with the initial deployment of IPv6, a set of mechanisms to ease and smooth the migration between IPv4 and IPv6 protocols were proposed which, consequently, also promoted the interoperability between both protocols. Although, these mechanisms were proposed in scope of IPv4 and IPv6, this section intends to overview the proposed solutions so that a set of lessons could be extracted and used as guidelines for the work developed in this thesis.

IPv6, designed as the successor of IPv4 protocol, defines a different packet format and addressing schemes (i.e., the size of addresses is increased from 4 bytes to 16 bytes). Entities only supporting the IPv4 protocol are not able to forward IPv6 packets and, in a similar way, entities only supporting IPv6 protocol are not able to forward IPv4 packets. As a result, IPv6 created a parallel networking environment that coexists with its IPv4 counterpart. The deployment of IPv6 has been slowly happening for a long time [26, 31], although in the recent years we have witnessed an increased traction on the IPv6 adoption[8].

During this transition period, where both IPv4 and IPv6 coexist simultaneously in the Internet, the interoperability strategies described in Section 2.3.2 have been used

[8]Google IPv6 Statistics - https://www.google.com/intl/en/ipv6/statistics.html

in the migration from IPv4 to IPv6 protocol, aiming to promote a smooth adoption of IPv6 by maintaining connectivity of both IPv4 and IPv6 networks and enabling the interoperability between IPv4- and IPv6-only hosts as well.

Following a dual-stack strategy, network entities (e.g., hosts and routers) run complete implementations of both IPv4 and IPv6 in parallel [48]. Such entities have the ability to handle both IPv4 and IPv6 packets and, thus, they can directly interoperate with IPv4 nodes using IPv4 packets and with IPv6 nodes using IPv6 packets. In addition, it enables IPv4 and IPv6 applications to operate in the same node, whenever dual stack is supported in the host [103]. Although most of the currently sold network equipments and hosts already support both protocols, to be possible to use such strategy in the whole Internet infrastructure, the legacy equipment supporting only IPv4 protocol would need to be replaced or updated.

To overcome such limitation of a dual-stack strategy, tunneling strategies started to be considered as well, as a mean to carry IPv6 packets over unmodified IPv4 routing infrastructure [48], acting as a bridge between IPv6 networks across incompatible IPv4 networks [103]. The tunnel endpoints can either be hosts or routers supporting both IPv6 and IPv4 protocols, on which the tunnel is established between: host-to-host, host-to-router, router-to-host or router-to-router. In the tunnel entry point an IPv4 header is added to the original IPv6 packet (i.e., the original IPv6 packet is encapsulated in an IPv4 packet). The packet can then be routed towards the tunnel exit point on which the IPv4 header is removed and the original IPv6 packet sent through the IPv6 routing infrastructure towards the original destination. To avoid explicit tunnel setup which introduces large management overheads, several approaches have been proposed to automate tunnel configuration. Proposals such as 6over4 [62], 6to4 [22], 6rd [34], and Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) [53] defines addresses that contain embedded IPv4 addresses, used to determine the source and destination IPv4 addresses when IPv6 traffic is tunneled across an IPv4 network. Another approach [36] relies on tunnel brokers which are responsible for automating tunnel management for requests originated from the users.

As IPv6 adoption becomes widespread, IPv6-only deployments are expected to occur. In order to maintain connectivity of residual IPv4 deployment (e.g., isolated IPv4 networking islands), it may be required to established tunnels over the IPv6 infrastructure. In the tunnel entry point an IPv6 header is added to the original IPv4 packet (i.e., the original IPv4 packet is encapsulated in an IPv6 packet). Dual-Stack Lite (DS-Lite) [35] uses IPv4 over IPv6 tunnels to forward traffic from a dual-stack host or Customer-Premises Equipment (CPE) provided with IPv6-only connectivity by the service provider (also called, DS-Lite Basic Bridging BroadBand (B4)), towards a carrier-grade IPv4-IPv4 NAT (also called, DS-Lite Address Family

Transition Router (AFTR)). Upon decapsulating packets coming from the tunnel, the AFTR performs IPv4-IPv4 NAT translation. B4 devices discover the AFTR via out-of-band mechanisms, manual configuration or using DHCPv6 options. Lightweight 4over6 approach [29] builds upon the mechanisms of DS-Lite, relocating the NAPT functionalities to the B4 device, in order to reduce the amount of centralized state at the AFTR. The approach proposed in [30] also leverages a tunneling IPv4 over IPv6 strategy where global IPv4 addresses are assigned to users, avoiding the need for carrier-side address translation. Although this last approach is in use in some existing deployments, future deployments of similar scenarios are recommended to use Lightweight 4over6 instead [30].

Finally, scenarios considering the communication between IPv4-only and IPv6-only applications/hosts (and vice-versa) are also addressed, with solutions exploring and proposing the translation between both protocols:

- The Stateless IP/ICMP Translation Algorithm (SIIT) [15] defines bidirectional algorithms for translating between IPv4 and IPv6 packets (and between ICMPv4 and ICMPv6 messages), not including IPv4 Options or IPv6 extensions headers (except the Fragment Header). Due to the stateless nature of this approach, packets between both protocols are translated based on the configuration of the translator and information contained within the packets being translated.

- Regarding the introduction of translators in the network path between hosts deployed in IPv4 and IPv6 architectures, IPv6-to-IPv4 Transport Relay Translator (TRT) [111] defines an intermediary entity that enables IPv6-only hosts to initiate and exchange TCP and UDP traffic with IPv4-only hosts. Following a stateful strategy, the TRT maintains two connections: one with the IPv6 host and another with the IPv4 host, relaying traffic between both connections.

- SOCKS-based IPv6/IPv4 Gateway [66] makes use of SOCKS protocol to enable IPv4/IPv6 traffic to be forwarded from the initiator applications to a relaying entity (i.e., the SOCKS server), which in turn establishes the correspondent IPv6/IPv4 connection with the destination node. This intermediary entity is, as in the previous case, responsible for relaying traffic between both connections.

- Bump-in-the-Host (BIH) [58], a successor and combination of the Bump-in-the-Stack (BIS) [13] and Bump-in-the-API (BIA) [65], proposes to allow IPv4-only applications running in dual-stack hosts to communicate with IPv6-only peers. The translation can be made at the socket API layer, with the translator intercepting IPv4 socket API function calls and, in turn, invoking the corresponding IPv6 socket API fucntion calls; or at the network layer, with packets being intercepted and converted using the algorithms defined in SIIT approach.

The migration from IPv4 to IPv6 proved to be far more complex than expected, with several mechanisms being proposed to mitigate such complexity. Most of the mechanisms proposed to ease and smooth the migration process have relied (i) in gateways supporting both protocols that (ii) followed a tunneling or translation approach to enable traffic to be sent across a different network architecture, (iii) while making the communication transparent to applications.

2.3.4 Interoperability mechanisms involving ICN architectures

As in the IPv6, interoperability between different ICN architectures and the current IP architecture started to be studied not only for enabling their coexistence in the same networking environment, but also as a mechanism to allow the incremental deployment of ICN architectures.

To our knowledge, [105] is one of the first works aiming to run the HTTP protocol on CCN, by proposing a HTTP-CCN gateway that converts HTTP requests and responses into CCN Interest and Data respectively (and vice-versa). In the ingress gateway towards the CCN network, HTTP messages are converted into CCN messages, allowing them to transverse the CCN network towards the egress gateway. In turn, by receiving the CCN messages, the egress gateway reconstructs the original HTTP messages, sending it towards the original HTTP server. This work supports the conversion of messages related with two of the most commonly used HTTP methods, the HTTP *GET* and *POST* methods. However, its operation scope is limited to the interoperability between HTTP and CCN protocols.

Versatile ICN Deployment Framework (VICN) [92] defines a framework that aims at facilitating the deployment of various ICN architecture instantiations, while enabling the interoperation among different ICNs. It leverages a separation of the data and control planes, where the data plane is extended to support ICN operations, processing received packets according to rules defined by the controlling entities. Regarding its capability to enable the interoperation among different ICN instantiations, VICN supports both tunneling and translation schemes. In the former, a tunnel between the access switch and a gateway switch is established so that packets from a different ICN instantiation can be transparently delivered over the tunnel. In the latter scheme, packets are translated between architectures so that any entity in the transversing network containing a cached copy can retrieve the content towards the user.

Another architectural framework, named Internames [76], defines a set of resolution services that allow names to be mapped not only to network locations but also to the protocol to be used to reach the current location of that name. In addition, it enables the coexistence of different network domains, which consists in an autonomous network using a local networking stack (e.g., IP and ICN). For that, it introduces dedicated nodes, called name-routers, that interconnect network domains and act as protocol proxies between name-oriented protocols operating on each network domain.

The H2020 POINT project [99] targets the deployment of IP-based applications over ICN-based architectures, aiming to provide IP-based services with a better performance than in standard IP networks. In order to maintain the user equipment unmodified, a gateway-based approach is proposed to preserve the IP interfaces towards the user equipment as well as IP applications and services. However, the scope of this gateway (called NAP) is strictly limited to IP and ICN architectures, although supporting handlers for specific IP-based protocols (e.g., HTTP and CoAP [41, 44]).

In [77], a TCP/ICN proxy capable of transparently carrying TCP traffic between TCP/IP endpoints over an ICN network is described and analyzed taking into account three alternative designs.

Finally, the Hybrid ICN solution [79] embedded ICN semantics into IPv6 packets, integrating ICN names in existing IPv6 headers and other ICN information carried as payload inside IPv6 packets. Such approach allows legacy IPv6 routers to forward packets based solely on IPv6 information while ICN-aware routers can process packets taking into consideration the ICN semantics.

2.3.5 Discussion

Proponents of the different architectures already provide infrastructure for some backwards compatibility with existing protocols of the current Internet architecture. Most of the existing work assumes the interoperability between different network architectures on a case-by-case basis or the interconnection of isolated islands belonging to a given network architecture over a different architecture (i.e., as an overlay).

However, with the advent of novel network architectures and, consequently, deployment of resources in these architectures, a key aspect for avoiding the creation of information silos is to allow entities to access the resources independently of the access network architecture. In such scenario, it would be possible to retain a single and integrated network, still perceived as the Internet as a whole. If a consistent and flexible framework was available to interconnect all Future Internet architectures, resources could be pooled together to enable full interoperability between multiple and changing architectures with no impact on the end hosts. To make this possible, such framework may build the necessary adaptation and control layers, focusing on the communication between entities deployed in different network architectures instead of overlaying traffic of a given network architecture over a different one.

2.4 Mobility management

With the development and proliferation of wireless and cellular access networks, a new paradigm in communications became available for users, enabling them to be mobile while maintaining connectivity. In addition, the variety of existing access technologies created an heterogeneous environment where each technology is able to provide superior capabilities (e.g., in terms of transmission rates, coverage or quality of service) in specific use cases. For example, Wireless Local Area Networks (WLANs) can be used indoors (such as, home or public facilities) for high transmission rates, while cellular networks can be used to provide wider coverage areas. Such heterogeneous environment stimulated the development of mobile nodes supporting multiple access technologies (e.g., smartphones), which are capable of selecting the most appropriate technology at each moment, providing an always best connected experience to the users [55].

Points of Attachment (PoAs) of different technologies (such as, WLAN access points or base stations in cellular networks) provide MNs with connectivity to the network. However, as users move, the signal strength received by the MN from PoAs varies due e.g. to the distance between the MN and the PoA. MNs may require to switch between different PoAs, resulting from e.g. degradation of the link quality, finding a PoA with better signal quality or being out of range of the current PoA. This procedure where the MN moves from one PoA to another is usually referred as handover and it is one of the most critical aspects in mobility.

During the handover procedure, link connectivity may be interrupted and, as a consequence, the communication channel between the MN and Corresponding Nodes (CNs) may also be interrupted. The time during which the MN is unable to send or receive packets due to link switching operations is called handover delay. The handover delay can have a negative impact on the performance of applications, due to which, depending on the time that the handover lasts, packet loss may occur and, in the worst case scenario, ongoing connections between the MN and CNs terminated. For example, the handover delay is a critical factor for time-sensitive and real-time application scenarios, on which packet losses may significantly affect the quality of experience of users. Also, most applications and services are not aware of the specificities of the link layer, not implementing any mobility-aware mechanism, and, therefore, after the handover to a new PoA, new connections may need to be re-established by applications.

As such, the goal of mobility management is two-fold: (i) to optimize handover procedures, reducing the handover delay so that its impact can be minimized; (ii) to provide seamlessly handover procedures, making handovers transparent for applications.

2.4.1 IP-based mobility management mechanisms

Whenever a node connects to a network, it acquires an IP address belonging to the network of the access router. If we consider the mobility of that given node, its IP address will change as it moves across different access network. Such change has consequences regarding IP reachability, requiring the reestablishment of ongoing connections by the MN. For example, a TCP connection is identified by a 4-tuple (source and destination IP address and source and destination port) and, therefore, a change on the IP address of the MN (i.e., the source IP address) will break ongoing TCP connections. To overcome this issue, IP-based mobility management mechanisms have been proposed in order to maintain the mobile node's reachability over the same IP address, allowing IP session continuity while on the move. Most of the presented proposals tackle IPv6 instead of IPv4, due to the capability of the hosts to be configured with several IPv6 addresses in the same network interface.

2.4.1.1 Centralized mobility management

Mobile IP (MIP) (both the MIPv4 [85] and MIPv6 [60]) is one of the first mechanisms developed by the Internet Engineering Task Force (IETF) to support node mobility. It defines that each MN is identified by an IP address that is independent of its location, called home address, and whenever away from its home network is associated with a second IP address, called Care-of Address (CoA), that identifies the current location of the MN. This enables the MN to change its PoA and still continue to be identified by its home address allowing seamless connection. MIP is also independent of the access technology of the current and the new PoA, i.e., it is suitable for mobility across homogeneous and heterogeneous networks. Two new functional entities are introduced in the network architecture: the Home Agent (HA) and the Foreign Agent (FA). While the HA is a router on the home network of the MN that maintains the current location of the MN and tunnels the messages towards the MN when it is away from its home network, the FA is a router on the visited network that routes the messages to the MN while registered. Specifically to Mobile IP for IPv6, there is no need to deploy the FA, which is able to operate in any location without support required from the local router [60].

When the MN is at the home network, packets to and from it are delivered by using standard IP routing. However, whenever the MN roams to a foreign network, the MN registers its new CoA with its HA to inform about its new location (i.e., binding procedure), establishing a tunnel between the HA and the CoA of the MN. In doing so, messages destined to the MN (i.e., addressed to its home address) are sent towards its HA which are then tunneled towards the its CoA and delivered to the MN. For messages sent by the MN there is no need to implement new routing mechanisms since the route of the messages to the destination is made according to conventional IP mechanisms. This routing approach, as depicted in Figure 2.9, is commonly named triangular routing.

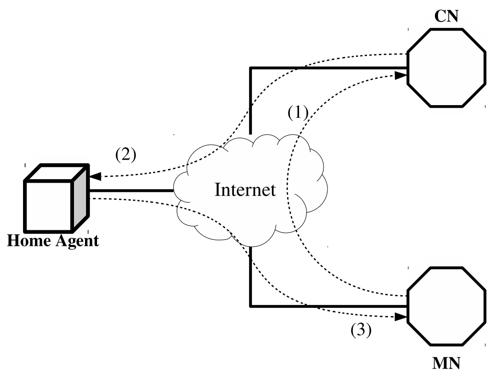


Figure 2.9: Triangular Routing in Mobile IP

Extensions to the base behavior of the protocol have been proposed to optimize the performance of MIP and, consequently, of the handover procedure. Fast Handovers in Mobile IPv6 (FMIPv6) [67] aims at reducing the handover latency, by enabling the MN to quickly detect that it has moved to a new subnet, providing the new access point and the associated subnet prefix information when the MN is still connected to its current subnet. Hierarchical Mobile IPv6 (HMIPv6) [17] aims at reducing the signaling overhead between the MN and its HA (and correspondent nodes) resulting from the handover procedure by introducing a new entity, called Mobility Anchor Point, which is located on the visited network and acts like a local HA.

Network-based mobility management solutions do not require the MN to be involved in the signaling exchanged with the home agent, with the mobility procedure being handled by a proxy mobility agent on the behalf of the MN. Proxy Mobile IPv6 (PMIPv6) [24], the network-based counterpart of MIP, proposes a mobility management solution within localized domains, providing mobility support for MNs without their direct involvement in the related signaling. Mobility entities in the network are responsible for tracking the movement of the MN, initiating the mobility signaling and setting up the required routing states. For that, it introduces two new network functional entities: the Mobile Access Gateway (MAG) and the Localized Mobility Anchor (LMA). While the MAG, deployed on the access link where MNs connects to, is responsible for performing the mobility management on behalf of the MN, the LMA is responsible for managing and maintaining the reachability towards the MN by establishing tunnels with the MAG.

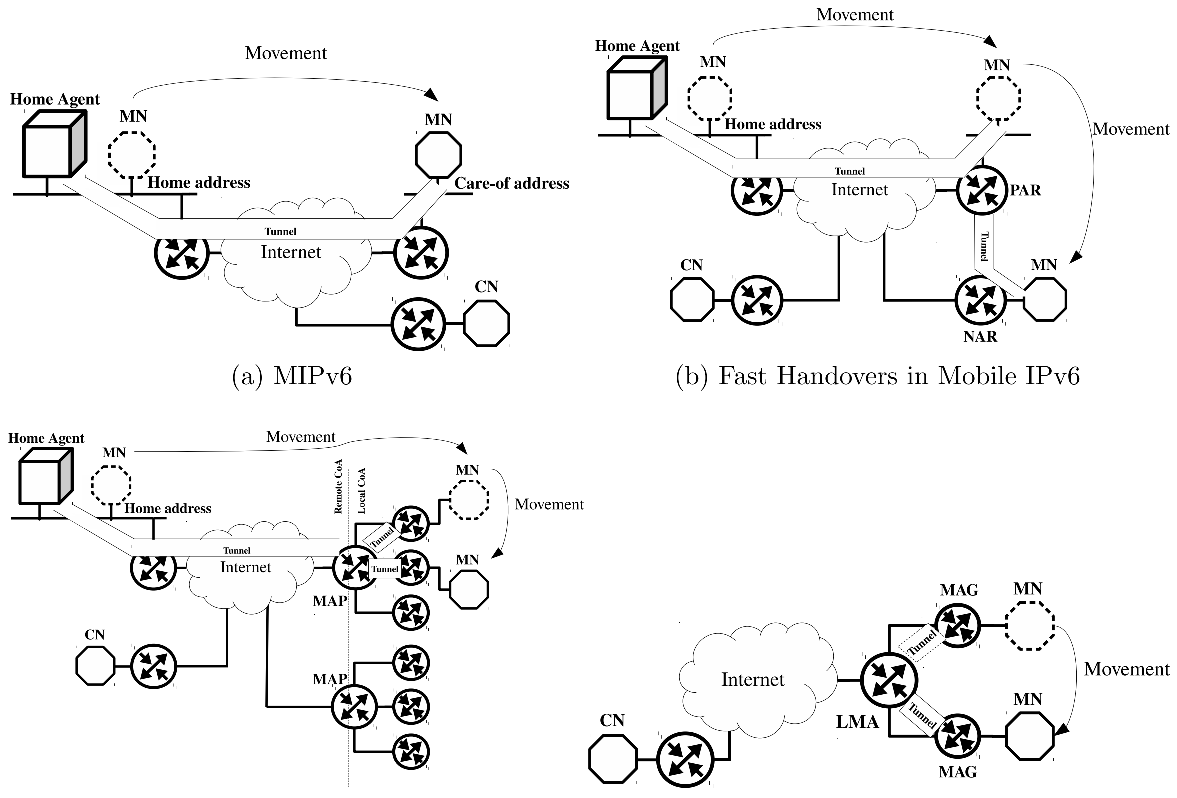
When the MN connects to an access network, the MAG on that link detects the attachment of the MN and it triggers the binding update procedure on the behalf of the MN in order to register it with the LMA. Upon successful registration, a bidirectional tunnel between the MAG and the LMA is established, used for routing the MN's traffic between the MAG and the LMA (i.e., used for routing packets from and to the MN). Packets destined to the MN are delivered to the LMA, which in turn forwards them towards the serving MAG via the established tunnel for the MN, to be finally sent towards the MN by the serving MAG. Whenever the MN switches its point of attachment from one MAG to another, a new tunnel is established between the new MAG and the LMA for the MN and the previous tunnel is teared down. Nevertheless, since the new MAG advertises the same home network prefix as the previous MAG, the MN is able to continue to use the same address that was obtained from the previous MAG.

An illustration of the previously described mobility management solutions is presented in Figure 2.10.

2.4.1.2 Distributed mobility management

The approaches of addressing mobility described above have in common the fact that they are centralized, with traffic requiring to go through a centralized anchor point in the network. This introduces several limitations regarding to non-optimal routing, low scalability and reliability aspects [23]. To mitigate the previous limitations, Distributed Mobility Management (DMM) [115] solutions started to be explored, where mobility anchors are deployed closer to the MNs (i.e., are relocated to the network edge). In doing so, MNs become capable of moving between mobility anchors, avoiding traffic to transverse a single mobility anchor far from the optimal route. DMM solutions can be categorized as partially distributed, where only the data plane is distributed, or fully distributed, where both the control and the data plane are distributed. Moreover, existing DMM solutions can be divided into two categories: (i) client-based; and (ii) network based solutions.

Several proposed solutions try to modify centralized mobility management



(a) MIPv6

(b) Fast Handovers in Mobile IPv6

(c) Hierarchical Mobile IPv6

(d) Proxy Mobile IPv6

Figure 2.10: Centralized mobility management solutions

approaches to work in a distributed way. Regarding client-based [51, 11, 12] and network-based [52, 49, 106, 18] DMM solutions, most of these proposals focus on distributing the operation of MIP and PMIP protocols, respectively. Other proposals, namely routing-based DMM solutions, leverage its operation on IP routing protocols to support mobility instead of tunnel establishments, being categorized as network-based solutions.

In both client-based and network-based DMM solutions, multiple mobility anchors are deployed in the network edge and, therefore, anchoring is distributed across multiple access networks. Instead of using a single IP address anchored at a central anchor entity, the MN configures additional IP addresses at each visited access network. Such approach allows the MN to use the locally-anchored address for new communications, while active communications can still use previous assigned IP addresses through bi-directional tunnels established between the MN and the corresponding mobility anchor in MIP-based DMM solutions or between the current and the corresponding mobility anchors in PMIP-based DMM solutions. In partially distributed approaches, a dedicated server (e.g., central mobility database), which tracks the MN and maintains mobility sessions of MNs, is responsible for managing the mobility of MNs.

2.4.2 Facilitating and optimizing handovers with IEEE 802.21

The IEEE 802.21 [1, 83] standard aims to define extensible media access independent mechanisms to facilitate and optimize handovers between heterogeneous technologies, including IEEE 802 (both wired and wireless) and cellular technologies. To achieve its purpose, it provides technology-agnostic means for mobility management entities to control and retrieve information from the access links, as well as it defines a set of messages for assisting mobility management entities during the different stages of the handover procedure. Note that it is out of scope of IEEE 802.21 how the actual handover procedure is executed, but it can be coupled with different mobility management mechanisms to facilitate and optimize their operation. The mechanism defined in IEEE 802.21 standard can be applied to both mobile and stationary entities. For example, due to the movement of mobile entities, changes in the current link conditions are likely to occur (e.g., signal strength starts to weaken, crossing a predefined threshold), triggering the execution of handover procedures. Regarding stationary entities, changes in the surrounding environment may also occur (e.g., malfunction of existing PoAs or deployment of new PoAs), which can also trigger the handover procedures to more attractive PoAs.

In order to operate and support the IEEE 802.21 mechanisms, each involved host and network entity must include a MIHF as a cross-layer function within their protocol stack. The MIHF acts as a middleware in the communication between the upper and the lower layers (i.e., MIH-users and Link SAPs respectively), abstracting the specificities of each technology from the upper layers. The MIH-Users are mobility management entities that makes use of the abstracting services provided by the MIHFs, while the Link SAPs are link-layer entities responsible for the specificities of each link-layer technology.

The services provided by the MIHFs can be categorized as:

- **Media Independent Event Service (MIES):** the MIES is responsible for notifying the upper layers about events from the lower layers, which can be originated from local or remote interfaces. These events are sent asynchronously to the MIHF, indicating changes in the state of the link layer or whether a pre-configured thresholds regarding specific link parameters (e.g., signal strength or throughput) is crossed.

- **Media Independent Command Service (MICS):** the MICS provides the ability to control, manage and get information from the link layers (e.g., order specific actions and parameters threshold configuration). The information retrieved by the command service is characterized by being synchronous information regarding the requested parameter. In addition, it also provides the ability to assist the handover procedure in its different stages (i.e., handover preparation, commit and complete stages).

- **Media Independent Information Service (MIIS):** the MIIS provides mechanisms to acquire, store and retrieve information about networks in the coverage area or within a geographical area, aiming to facilitate handover procedures. In doing so, it allows the mobile devices and network entities to discover information, such as knowledge of security information, supported channels, cost per use, networks categories and QoS supported, which helps in the selection of appropriate networks during handovers.

The IEEE 802.21 standard also defines a protocol, the Media Independent Handover (MIH) protocol, for MIHFs to communicate with one another. The communication between peer MIHFs is essential to enable the optimization of handovers between heterogeneous networks, allowing mobility management entities on each involved entity to assist in the handover procedure of the mobile devices. This protocol defines the format of the messages exchanged between peer MIHFs, as well as the mechanisms that support their transport and reliable delivery of messages to their destination.

Several amendments, namely IEEE 802.21a[3], IEEE 802.21b[2], IEEE 802.21c[4] and IEEE 802.21d[5], have been proposed to enhance the capabilities of IEEE 802.21 standard mechanisms. These amendments target, respectively, security extensions, extensions for supporting handovers with downlink only technologies, optimized single radio handovers, and multicast group management.

2.4.3 Mobility in evolutionary SDN-based architectures

The separation of the control and data planes leveraged by SDN, allowing the network behavior to be programmed from a central entity, has motivated the use of SDN in mobility management solutions aiming to optimize and improve their operation. This separation of the data and control planes fits into the approach taken by several mobility management solutions, on which a centralized entity is responsible for the control plane, managing the mobility procedure of MNs. In terms of mobility management, the SDN controller can act as the dedicated entity for mobility procedures, managing the mobility procedure of MNs and (re)configuring the network to cope with the MNs movement. It is responsible for (re)configuring the forwarding rules not only in the mobility anchor but also in intermediary switches in the network. Whenever the MN attaches to a new anchor point, the network controller is notified about such event, which in turn reconfigures the network to redirect the traffic related to the MN towards its new location.

To achieve this goal, two different approaches can be found in the state of the art. On one hand, existing mobility management protocols are enhanced to cooperate with SDN mechanisms. On the other hand, new mechanisms are defined based solely on SDN without requiring existing mobility management protocols.

In what regards the former, solutions such as [87], [64, 91] and [102, 80] enhance, respectively, MIP, PMIP and DMM approaches with SDN mechanisms to ease in routing path configuration procedures. Taking as example the solution proposed in [64], it defines two operation modes for an OpenFlow-based PMIPv6 solution: *OPMIPv6* where the LMA and the MAG are respectively instantiated in the OpenFlow Controller and OpenFlow switches, and *OPMIPv6-C* where both the LMA and the MAG are instantiated in the OpenFlow Controller. In OPMIPv6 operation mode, PMIPv6 signaling is exchanged between LMA and the MAG for notifying the attachment and forwarding the assigned home network prefix, while OpenFlow signaling is used to set up the routing path avoiding the need for IP tunnels. In OPMIPv6-C, each of these operations are handled by the MAG and LMA functions in the OpenFlow Controller, using only the OpenFlow protocol for such purpose.

Regarding the latter, in the solution presented in [50] the network (re)configuration consists in a combination of translation and forwarding rules on the anchor points. The previous anchor points of the MN are configured to rewrite the IP destination address of incoming packets destined to the MN with the last known address of the MN, while the current anchor point of the MN is configured to perform the reverse translation, restoring the old IP address of the MN. In doing so, whenever traffic from an anchored flow reaches any of the previous anchor points of the MN, it is redirected towards the current location of the MN. Although this solution does not involve any IP tunnel, its behavior is mimicked. Other solutions [95, 20, 107] tackle the problem by updating the routing path after the handover procedure to allow packets destined to the MN to be directly redirect towards the current location of the MN, eliminating the burden of requiring to go firstly to the previous location.

2.4.4 Mobility in clean-slate ICN-based architectures

By focusing the communication on names (*what*) instead of network addresses (*where*), ICN architectures try to move away from any location dependencies, and, therefore, mobility is handled in ICN in a different way than in the current IP architecture. In ICN, mobility can be tackled in terms of consumer and/or provider mobility, on which the former aims to allow consumers of content objects to move without disrupting connectivity and the latter to allow producers of content objects to move without disrupting content availability [101]. This thesis focuses on improving the efficiency of consumer mobility.

Communications in ICN are commonly characterized by being consumer-driven. Consumers follow a pull-based communication model to fetch a given content object, and by being connectionless. As such, mobility of consumers is intrinsically supported by such architectures. More specifically, the consumer just needs to re-issue its interest on the desired content after the handover or when the previous requests time out. Regarding the ICN instantiations studied in this thesis, after moving to a new PoA, consumer mobility can be achieved by:

- in NDN, consumers re-issue *Interest* messages for any previously sent request for a content object not yet satisfied with the purpose of recreating the reverse path back to its new location [73, 100]. If the new and the old paths cross, the *Interest* message can be: (i) retrieved by an intermediary entity containing a cached copy of the content (i.e., while satisfying the *Interest* messages pending on the old location of the mobile consumer, content can be cached in intermediary entities) [14]; or (ii) aggregated with the old request in the correspondent PIT entry (i.e., the *Interest* message sent from the old location was not yet satisfied) not being propagated further [114].

- in PURSUIT, consumers resubscribe to the content being accessed. This triggers the *Rendezvous* and the *Topology Nodes* to recalculate forwarding identifiers between the producer and the new location of the mobile consumer [109, 100]. The new forwarding identifier is sent to the content producer, so that messages can start being sent towards the new location of the mobile consumer.

- in ETArch, the mobile consumer re-registers itself in the new PoA, after which it re-attaches to the *DTS workspace* where the content is being provided. This leads the DTSA to reconfigure the network in order to extend the *DTS workspace* towards the new location of the MN, so that messages can start being forwarded towards the new location of the mobile consumer.

However, the previous approaches share some drawbacks when considering time-sensitive applications (e.g. live streaming video and audio and/or video conference): (i) the delay introduced by the convergence mechanisms on each network architecture to enable the mobile consumer to restart the reception of the requested contents may lead to content loss; and (ii) the already requested contents may still be sent towards the old location of the consumer, leading to an unwanted utilization of bandwidth and network resources. To avoid it, the mobile consumer may need to cancel its interest for the contents being accessed before the handover (PURSUIT and ETArch have messages for unsubscribing, but NDN does not have messages for canceling its interest on a given content) and request them again after the handover, but such strategy may also lead to content loss (i.e., content may be loss during the time between both operations).

2.4.5 Discussion

Mobility in both IP and ICN architectures has been a subject of study by the research community to enable MN to maintain connectivity after the handover procedure. However, the main scope of the research on mobility in both IP and ICN architectures is commonly centered on how to enable the MN to maintain connectivity after the handover procedure, disregarding any optimization that could be leveraged by taking into account context information from the link-layer of both mobile nodes and networking entities (such as, PoAs). This context may provide useful information to decision making entities, such as the detection of handover opportunities and selection of the best available PoA for a given MN to move to. For example, the detection of an imminent failure of a PoA may trigger the handover of a (group of) MN(s) towards a different PoA, or simply the detection that a given MN has on its range a PoA with better signal strength than its current PoA may trigger the MN to move to that MN. Also, this allows mobility management entities to (re)configure the state of network entities before and after the handover occurs with the purpose of mitigating the effects caused by the handover itself.

2.5 Concluding Remarks

In this chapter, we have addressed the key concepts and technologies associated to the different areas focused by the work developed under this thesis. Future Internet architectures, both evolutionary (e.g., SDN-based) and clean-slate (e.g. ICN), are introduced as the starting point for the discussion of the interoperability and mobility, aspects that compose the core research of this thesis. These aspects are then further explored, with the description of the most relevant related work, followed by a brief discussion in an attempt to introduce to the most important motivations that paved the way to the thesis work.

Chapter 3

Binding the Future Internet Architectural Landscape

Given the previously identified set of potential deployment scenarios of a new network architecture in the existing networking environment and the possible interoperability strategies between different architectures, this chapter focuses on achieving an integrated solution for supporting the interoperability among different network architectures in a networking environment where multiple architectures coexist in parallel.

3.1 Introduction

Due to the disruptive nature of clean-slate network architectures when compared with evolutionary architectures, the deployment of such architectures is not envisioned as an easy process, since they might impose architectural changes ranging from the network up to the application layer. Nevertheless, this thesis considers the existence of different network architectures coexisting with one another in parallel [39], a critical aspect for the incremental deployment of new network architectures without impacting the existing networking landscape. The deployment of new architectures can then be done incrementally through globally connected islands, either as an overlay over existing networks or directly connected through dedicated links in specific business scenarios.

However, with multiple network architectures available simultaneously, a question is risen on how to enable a clear path for resource providers to use those novel architectures, so that resources are not restricted to a single network architecture in an island. On one hand, providers might not invest on the replication of resources over different architectures without a significant advantage on sight. On the other hand, an increase of complexity on the users side will impact usability. For a functional heterogeneous Future Internet, interoperability mechanisms between the different architectures are required to enable users, applications and network mechanisms to access resources independently of the network architecture of the communication endpoints.

In this chapter an interoperability framework, named Future Internet Fusion (FIFu), is presented that enables different architectures to coexist and to exchange information between one another without bounds to any particular instance. Such framework simplifies the provisioning of resources in network architectures other than the one where they are physically deployed in, making the process manageable for the resource providers and facilitating the incremental deployment of new architectures without creating information silos.

3.2 Problem and Background

This section introduces the scenario that motivated this work, followed by the identification of the key interoperability challenges.

3.2.1 Reference Scenario

The reference scenario (depicted in Figure 3.1) considers the access to a given resource by entities deployed in three different network architectures, each developed by taking into consideration different design choices. More specifically, the considered network architectures differ in aspects such as (i) design orientations; and (ii) communication paradigms (as discussed in the next section).

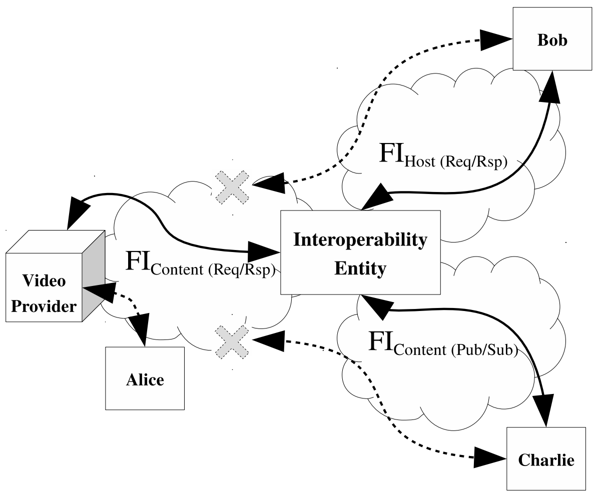


Figure 3.1: Reference interoperability scenario

Alice, Bob and Charlie are watching the video stream of a sports event broadcasting at the stadium. The network architecture deployed at the stadium (referred to as $\text{FI}_{\text{Content(Req/Rsp)}}$) is content-oriented and operates on a request/response communication paradigm.

Although Alice is at the stadium viewing the football match, she wants to e.g. view the replay of a goal and, since her device supports the same network architecture than the one provided by the stadium (i.e., $\text{FI}_{\text{Content(Req/Rsp)}}$), she can obtain the video stream directly from the video server by addressing the content itself and requesting it via polling mechanisms (i.e., using a request/response communication paradigm). In contrast, Bob and Charlie are at home and connected to the Internet through a host-oriented network architecture that operates based on polling mechanisms (referred to as $\text{FI}_{\text{Host(Req/Rsp)}}$) and a content-oriented publish/subscribe network architecture (referred to as $\text{FI}_{\text{Content(Pub/Sub)}}$) respectively. In both cases, the devices of Bob and Charlie, and the networks they are connected to, support a different network architecture than the one at the stadium and, therefore, they are unable to directly fetch the video stream broadcasting at the stadium, due to the incompatibilities between their network architectures and the one at the stadium.

As such, Bob and Charlie need to request the video stream through mechanisms and protocols supported by their devices and networks. Bob requests the video stream using protocols supported by $\text{FI}_{\text{Host(Req/Rsp)}}$ and Charlie using protocols supported by $\text{FI}_{\text{Content(Pub/Sub)}}$, both from an intermediary entity (i.e., the interoperability entity) that is able to convert the messages into a $\text{FI}_{\text{Content(Req/Rsp)}}$ format (and vice versa). This interoperability entity acts on behalf of Bob and Charlie to obtain the video stream and as the video server in the network architectures of both Bob and Charlie, behaving as an adaptation layer between the different network architectures.

Even if part of the communication does not exploit all the advantages associated with the networking approach deployed at the stadium, Bob and Charlie can still access the video stream through a different architecture, and thus it is possible to retain a single and integrated network, still perceived as the Internet.

3.2.2 Interoperability Challenges

The design choices taken by each network architecture can hinder their interoperation due to incompatibilities in specific operational aspects, making interoperability impossible without an intermediary entity. Next, a set of potential hindering aspects when interoperating between different architectures is discussed.

3.2.2.1 Different design orientation

Today's IP network architecture is oriented around hosts and end-to-end principles. However, other network architectures are centering their network layer on different elements, such as services, content, users, or even “anything”. By shifting the design orientation, different operational principles may be implemented by the network architectures, changing how networks and applications are designed, developed and deployed.

Challenge 1: *How can network architectures defined by a different set of operational principles interact with each other in order to exchange information?*

3.2.2.2 Different communication paradigms

According to how information is obtained, communication paradigms may be categorized as *pull* or *push.* In the *push* paradigm the information is sent towards the destination without having a prior request, which contrasts with the *pull* paradigm where an explicit request for the information needs to be sent. Moreover, *pull* communication paradigms can be sub-categorized into *publish/subscribe* or *request/response*, which differ on how the desired information is requested and delivered. While the former only needs to request the information once (i.e., subscription) and afterwards information is sent to the requester whenever an update occurs, the latter needs to send an individual request for each information retrieval.

Challenge 2: *How to adapt communications between architectures that follow different paradigms for obtaining information?*

3.2.2.3 Different addressing schemes

As each network architecture addresses resources in different ways, an heterogeneity of addressing schemes with different semantics and encodings is created. Addresses may represent geographic locations, administrative domains, hosts, services, contents, users, among others; may be encoded with a fixed or variable length; may be hierarchical or flat; may be self-certified or not; etc. Moreover, while some network architectures addresses the resource directly at the network layer (e.g., content or services), others address the resource itself at the upper layers in which the network-level address represents just a hint on how to reach the resource (e.g., the host that has the resource).

Challenge 3: *How to map addresses between architectures when these are scattered among multiple layer protocols?*

Challenge 4: *How to map addresses with different meanings?*

Challenge 5: *How to convert between addresses with different encodings?*

3.2.2.4 Different network protocols

The network protocol is the operative tool of the network architecture at the network layer and, therefore, for an equipment to support a given network architecture it needs to support the correspondent network protocol. As such, for an equipment to support a new network protocol it may be required its update or replacement. Otherwise, whenever a message from an unknown network protocol is received by the network equipment, the most common decision is to discard the message, failing to deliver it to its destination.

In addition, the network protocol of a given network architecture may implement features from one or more upper layer protocols on another architecture. For example, some functions implemented in upper layers of the IP architecture are already addressed at the network layer in ICN approaches (e.g., security and content integrity). In this way, layers above the network layer deployed over a given network protocol may not be directly transposed when applied to a different network protocol.

Challenge 6: *How to convert messages from one network protocol to another?*

Challenge 7: *How to maintain backwards-compatibility while converting between network protocols?*

These differences between network architectures hinder the compatibility with one another. As a result, entities deployed in a given network architecture may be restricted to the resources available in that architecture, not being able to access resources available in a different architecture. Thus, mechanisms to enable the interoperability and the exchange of information between entities in different architectures are required to allow a functional heterogeneous Future Internet. The discussion above also highlights that the problems are deeper than a solution handled at a router level.

3.3 FIFu: Enabling Future Internet Interoperability

The need for a heterogeneous Future Internet, where different network architectures coexist and interoperate, acted as the main driver for the development of the Future Internet Fusion (FIFu) framework. This framework provides an infrastructure that enables different network architectures to exchange information and, therefore, to interoperate with one another. In doing so, it contributes to the reduction of resource provider deployment times and it allows a global access to resources.

One of the key requirements that were on the basis of the presented framework is that the interoperability mechanisms must be transparent for both the communication endpoints and the network entities on each network architecture. In other words, both the endpoints and network entities must perceive the exchanged messages as being originated by an entity in the same architecture and, thus, unaware that messages are from/to a different network architecture. In doing so, such framework is expected to leverage the development, implementation and evolution of each network architecture without imposing any restriction in terms of interoperability.

Inspired by Software Defined Networking [68] concepts, the FIFu framework (depicted in Figure 3.2) consists in a two-layer framework composed by: (i) the adaptation layer; and (ii) the intelligence layer. The adaptation layer connects Future Internet network islands operating under different network architectures. This layer is composed by a set of interoperability entities, called *Future Internet eXchange Points* (FIXPs), that are responsible for converting messages from/to different network architectures. The adaptation layer entities are under the control of logically centralized entities, called *Future Internet Controllers* (FICs), that compose the intelligence layer. The FICs are responsible for configuring, managing and supporting the FIXP operation, as well as for creating, storing, managing and maintaining the mappings of resources on each network architectures.

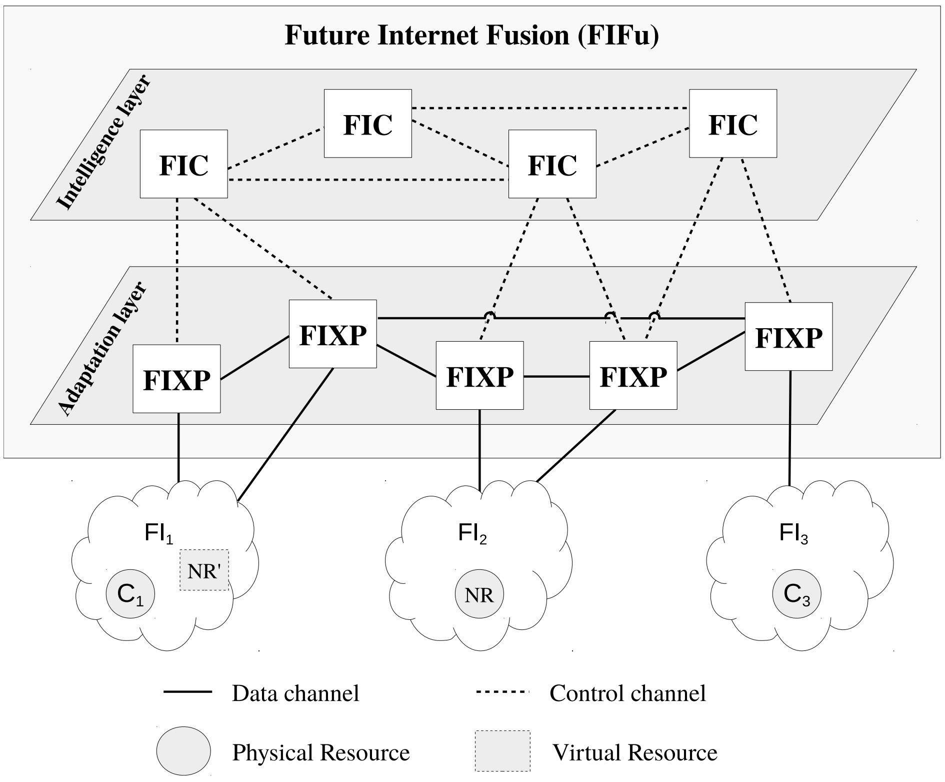


Figure 3.2: FIFu framework overview (NR is accessible from FI_1 but not from FI_3)

For a given resource to be available in a network architecture (besides the one it is deployed in), it has to be accessible using addresses compatible with the protocols supported by each network architecture. In the architectural environment presented in Figure 3.2, the resource *NR* in FI_2 can be accessed by clients deployed in FI_1 (i.e., *C1*) but not from clients deployed in *FI₃* (i.e., *C3*). The reason is that in *FI₁* a virtual representation of the resource *NR* (i.e., *NR'*), accessible through addresses compatible with the protocols supported by that architecture, is available. In turn, in *FI₃*, the client *C3* cannot access the resoure *NR*, since *FI₃*-compatible addresses are not yet available (i.e., no virtual representation is available).

Within the FIFu framework, Uniform Resource Identifiers (URIs) [19], as exemplified in Figure 3.3, are used to represent resource identifiers, which consist in a sequence of characters that identifies an abstract or physical resource. An URI provides all the required information (or hint(s) that lead to the required information) on how to access the identified resource, such as the protocol and correspondent address to be used, along with lower layer information/addresses when required. For example, "*http://atnog.av.it.pt*" and "*ndn:/atnog.av.it.pt*" are URIs that may be used for identifying resources, namely a web page in IP and a content object in NDN architectures, respectively.

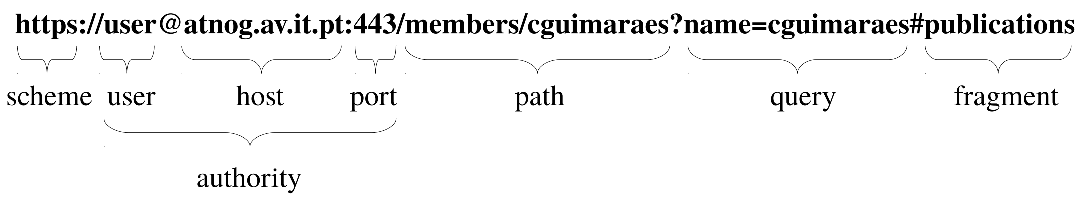

https://user@atnog.av.it.pt:443/members/cguimaraes?name=cguimaraes#publications
scheme
user
host
port
path
query
fragment
authority

Figure 3.3: URI component parts

In the presented framework, a given resource is identified by a single original URI, which identifies how the resource is accessed from the network architecture where it is physically deployed, and multiple foreign URIs, which identify how the resource is accessed from the remaining network architectures via regular and well-known protocols of each architecture. The foreign URIs enable existing applications, services and network mechanisms to view resources as belonging to their network architecture, even if not physically deployed in that architecture. As referred in the previous example, these correspond to virtual representation of the resource on the different network architectures. This strategy enables not only backwards compatibility with existing mechanisms, services and applications but also allows mechanisms of each architecture (e.g., caches, name resolution services, search engines, *etc.*) to be applied to the foreign resources in a transparent way (i.e., independently of the architecture where they are deployed), while supporting an independent and continuous evolution of each architecture.

In the remainder of this chapter, the terms "identifier" and "URI" will be used interchangeably. Moreover, the original network architecture will be referred as the network where resources are actually deployed in, while the foreign architecture will be referred as the architecture of the entities wishing to access those resources. The destination network architecture will be used for referring to the network architecture where the eventually converted messages are sent to, which can either be the original or the foreign network architecture.

3.3.1 Adaptation Layer

The adaptation layer is composed by the *Future Internet eXchange Points*, intermediary entities responsible for providing interoperability between different network architectures, allowing the communication between endpoints deployed in different architectures. As such, the FIXPs, acting as gateways, emulate the communication endpoints of the original network architecture on the foreign architecture and vice versa, simultaneously behaving as the destination and source of the messages (as represented in Figure 3.4).

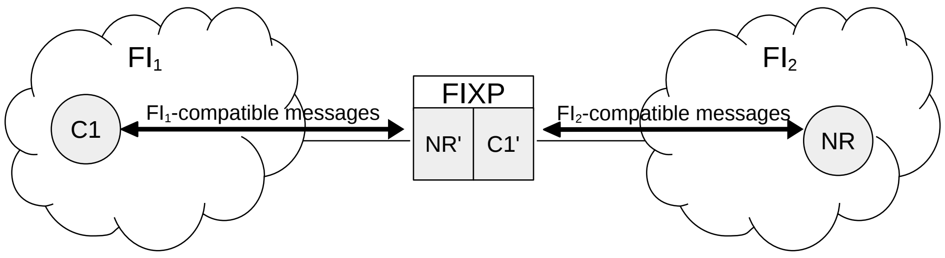


Figure 3.4: FIXP message conversion

As part of the adaptation process, FIXP entities receive messages from a given network architecture and convert them in order to be compatible with the destination architecture. As such, for each protocol, FIXPs are configured with information about the packet format and the semantic meaning of each packet field, allowing them to recognize and understand the existing protocols on each network architecture.

The provisioning of this information to the FIXP is conducted by the intelligence layer (i.e., the FIC), without requiring service disruptions or equipment replacement. For each supported protocol of each network architecture, the FIC deploys in the FIXP the running instance that will enable the latter to act as a source and destination endpoint in what concerns that protocol (i.e., the architecture protocol endpoint). Each architecture protocol endpoint is responsible for the following tasks:

- understanding the corresponding protocol/architecture;
- being able to receive, identify, process, create and send messages related with that protocol/architecture;

- extract the content and related metadata (e.g., security level and QoS) from the received message as well as to apply them to the created messages;

After configuring the FIXP with the architecture protocol endpoints, converting messages from one network architecture to another can be divided into three major operations: (i) address translation; (ii) signaling adaptation; and (iii) content adaptation. The processing workflow of incoming messages in the FIXP is presented in Figure 3.5.

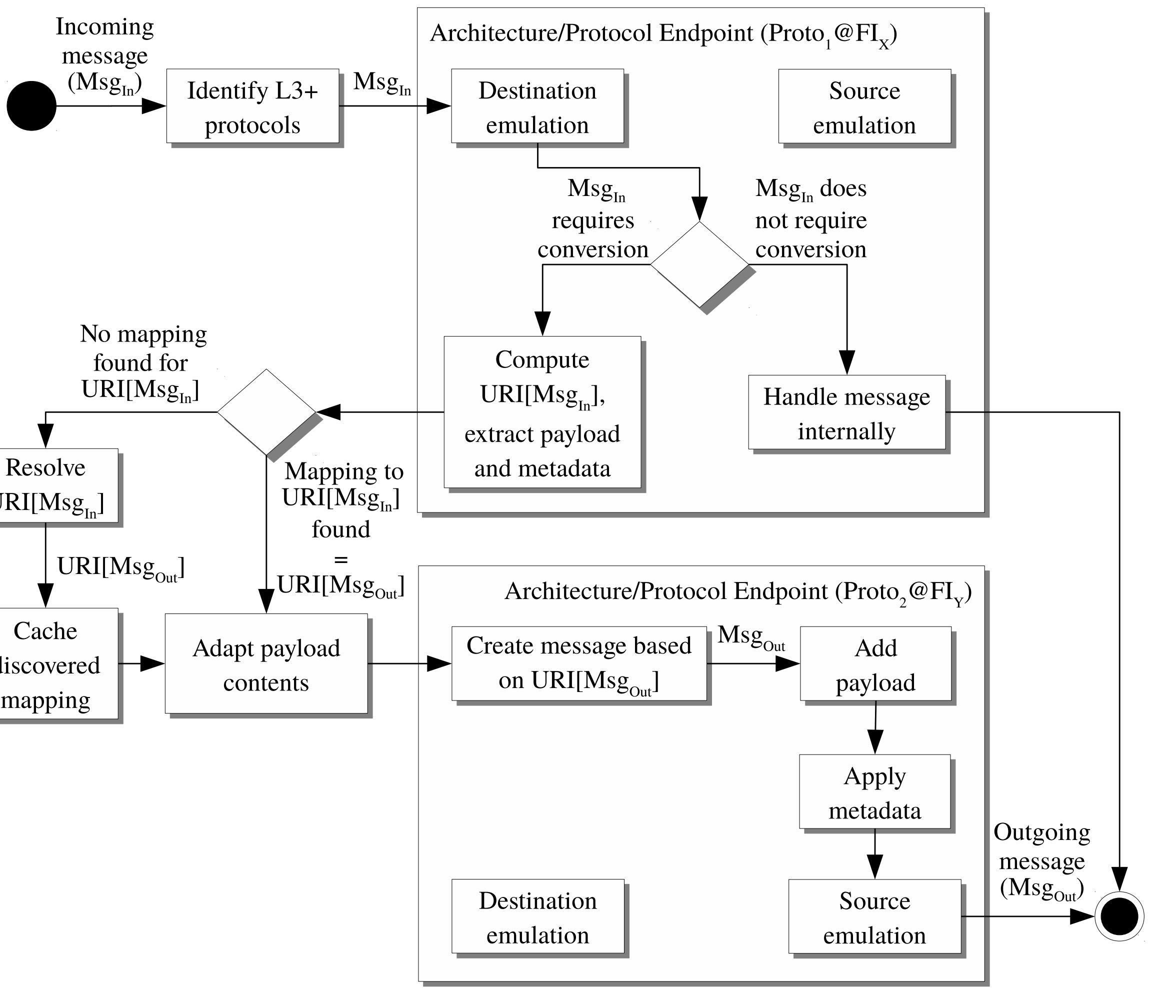

Incoming message (Msg_{In})
Identify L3+ protocols
Msg_{In}
Architecture/Protocol Endpoint ($Proto_1$@FI_X)
Destination emulation
Source emulation
Msg_{In} requires conversion
Msg_{In} does not require conversion
Compute URI[Msg_{In}], extract payload and metadata
Handle message internally
No mapping found for URI[Msg_{In}]
Resolve URI[Msg_{In}]
Mapping to URI[Msg_{In}] found = URI[Msg_{Out}]
URI[Msg_{Out}]
Cache discovered mapping
Adapt payload contents
Architecture/Protocol Endpoint ($Proto_2$@FI_Y)
Create message based on URI[Msg_{Out}]
Msg_{Out}
Add payload
Apply metadata
Destination emulation
Source emulation
Outgoing message (Msg_{Out})

Figure 3.5: FIXP message processing workflow

Whenever a new message is received, the FIXP inspects it, aiming to identify the L3 and above protocols (i.e., L3+ protocols) that compose the received message. Based on the protocols that compose the received message, it is redirected to the respective architecture protocol endpoint. In turn, since the architecture protocol endpoint emulates the destination endpoint of the received message, it verifies if the message needs to be converted and sent towards the destination architecture. If not, the message is handled internally according to the corresponding protocol stack. Otherwise, the URI of the received message is computed (i.e., a unique identifier that comprehends all the spread resource identification information) and the associated payload and metadata extracted.

In the following step, the FIXP searches on its cache for a mapping related with the computed URI (i.e., URI[Msg_{Out}]), which will then be used to generate the equivalent message in the destination architecture (i.e., Msg_{Out}). If no mapping is found in the cache, the FIXP resolves the URI[Msg_{In}] with the FIC infrastructure in order to find its corresponding in the destination network architecture, being cached therein to avoid future resolutions. After discovering the destination network architecture and protocol to use (based on the scheme part of the URI[Msg_{Out}]), the payload content is adapted if required. All this information (i.e., URI[Msg_{Out}], the payload and metadata) is internally forwarded to the respective architecture protocol endpoint. The architecture protocol endpoint creates the protocol message using the URI[Msg_{Out}], after which the payload is added and the metadata applied using the specific protocol procedures. Finally, the message is sent towards the destination network architecture.

In order to avoid the communication endpoint to be aware of any interoperability mechanism, and thus to fully achieve backwards-compatibility with existing applications, contents may be required to be in compliance with the destination network architecture. An example of this need is the web browsing use case. Taking the example from Figure 3.2 and assuming that the client *C1* is a web browser and the network resource *NR* is the homepage of a given website, the FIFu framework enables the web browser to fetch the homepage, which is deployed in a different network architecture. However, for the user to be able to continue navigating in the web page, URIs contained therein must be compatible and supported by both the user's web browser and the network architecture (i.e., FI_1). As such, if contents are not adapted to be in compliance with the destination architecture, subsequent requests for other resources in the web page may be discarded due to (i) the web browser or network stacks in the user device do not support the protocol required by the URIs; or (ii) the application and the device support the required protocol, but the network architecture to which it is connected does not. In both cases, the user is unable to continue navigating on that web page.

In this context, to avoid the need for resource replication explicitly for each network architecture, making the process transparent (or at least manageable) for the resource provider, the presented framework needs to be intrusive in the sense that the adaptation layer (i.e., the FIXPs) may need to change existing identifiers inside the payload of each message. As such, during message conversion, depending on the content type, the FIXP may inspect the message payload looking for resource identifiers (e.g., links in HTML pages) which are then replaced by compatible identifiers with the destination network architecture if required.

3.3.2 Intelligence layer

The intelligence layer controls and supports the FIXP operation. It is composed by the *Future Internet Controllers*, which hold two sources of knowledge: (i) a list of the FIXPs deployed throughout the Internet; and (ii) a repository of mappings between identifiers of resources in different architectures used by FIXPs to convert messages between protocol stacks. Following this approach, the FIFu framework aims at leveraging a more flexible and faster approach to accommodate new network architectures and how they are interconnected without requiring to manually configure each FIXP.

The FICs control, assist and synchronize the operation of the adaptation layer. Their tasks include resource monitoring and configuration of the architecture protocol endpoints supported by each FIXP. To achieve their goal, the FICs are able to discover the capabilities of the underlying FIXPs (such as their hardware specifications - e.g., CPU, memory, network interfaces, etc), the currently supported network architectures/protocols, to which network architecture(s) they are connected to and to obtain statistics about the traffic being converted. This information can be acquired using two strategies: (i) by polling each FIXP periodically; and (ii) by subscribing a set of events (and/or pre-configuring thresholds) on the FIXP which, whenever they occurred (and/or are crossed), generates a notification towards the FICs.

For example, the FIC can configure the underlying FIXPs to periodically notify about the bandwidth usage on each polling period or to trigger a notification whenever the CPU or memory usage crosses a given threshold. This information can then be leveraged to (re)configure the FIXPs, optimizing their operation and allowing a dynamic adaptation of the network according to its current conditions, as well as to augment the capabilities of the FIXP with the understanding of new protocols by installing new architecture protocol endpoints.

FICs also act as a hierarchical distributed mapping resolution system, managing the identifiers of the resources on each network architecture. As such, they are responsible for creating, storing, managing and maintaining the mappings between resource identifiers on different network architectures, while providing a resolution service to retrieve the mapping related information whenever requested.

The logical hierarchical organization of the FICs, regarding their operation as a mapping resolution system, is divided into three levels, as depicted in Figure 3.6:

1. **Root level:** is the highest level of the resolution system, being queried to obtain the network architecture level FIC;

2. **Network Architecture level:** creates and maintains foreign URIs belonging to the scope of network architecture they are responsible for and manages their distribution across the URI level FICs; and

3. **URI level:** the lowest level, used for storing the mappings between URIs on different network architectures.

Nevertheless, information of the different levels can be cached and distributed across different FICs, easing its scalability, load-balancing and redundancy.

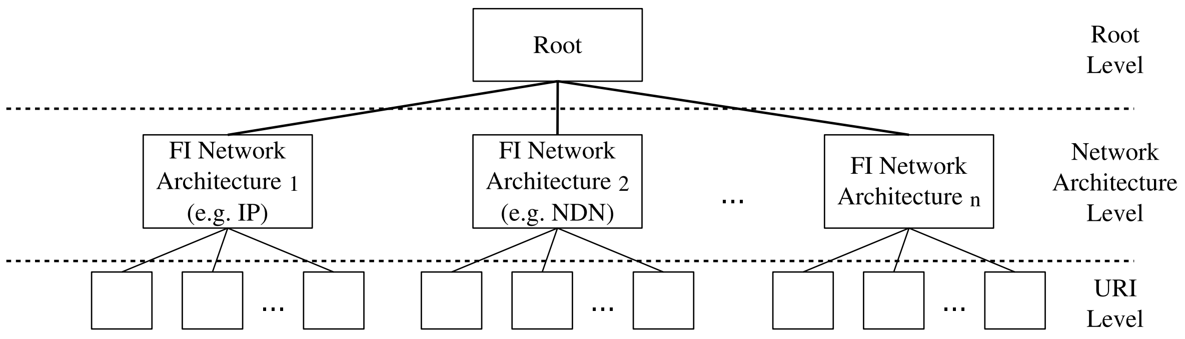

Root
Root Level
FI Network Architecture 1 (e.g. IP)
FI Network Architecture 2 (e.g. NDN)
...
FI Network Architecture n
Network Architecture Level
URI Level

Figure 3.6: FIC (mapping system) logical topology

Even though the resource provider can preemptively register mappings for its resources, foreign URIs can be automatically generated by the FIC and mapped to its correspondent URI on the original network whenever needed. Upon the reception of resolution requests for a given URI, if a mapping does not exist, the correspondent *Network Architecture level* FIC responsible for the foreign architecture (which resolution request was made) generates the URI to be used in the foreign architecture. The generated URI is then delivered to the requester and stored in the appropriate *URI level* FICs.

FIXPs (or other FICs) can then resolve foreign URIs into their original form and vice versa. This resolution can be either iterative or recursive in a similar way to DNS operation modes. In iterative queries, if the FIC does not have the required information, it replies with a referral to another FIC that may have the information. In turn, in recursive queries, the FIC replies with the requested mapping or an error message. Therefore, if the FIC does not have the required mapping information, it recursively requests other FICs until it gets the desired information or until the query fails.

Figure 3.7 presents the signaling used by the FIXP to resolve foreign URIs into their original form and vice versa.

For requesting the resolution of a given foreign URI into its original form, the FIXP requests the resolution of the foreign URI towards its local FIC (message 1 in

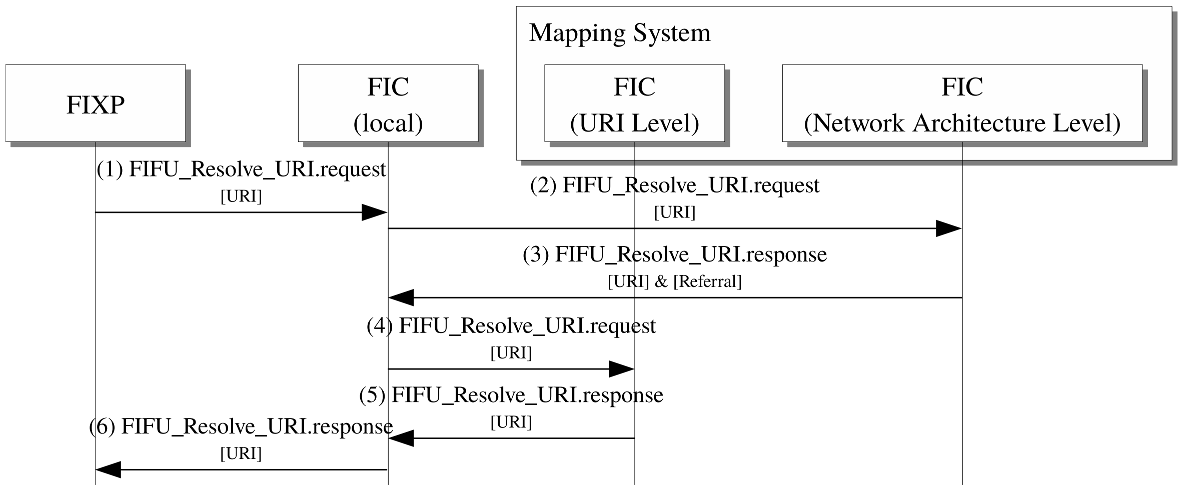

Mapping System
FIXP
FIC (local)
FIC (URI Level)
FIC (Network Architecture Level)
(1) FIFU_Resolve_URI.request [URI]
(2) FIFU_Resolve_URI.request [URI]
(3) FIFU_Resolve_URI.response [URI] & [Referral]
(4) FIFU_Resolve_URI.request [URI]
(5) FIFU_Resolve_URI.response [URI]
(6) FIFU_Resolve_URI.response [URI]

Figure 3.7: Resolving URIs inside FIFu framework

Figure 3.7), which, in turn, requests the resolution towards the *Network Architecture level* FIC (message 2 in Figure 3.7). If the *Network Architecture level* FIC is not known, the request is forwarded towards a *Root level* FIC that will then return the referral to it. In this case, the *Network Architecture level* FIC is selected based on the scheme of the foreign URI. The *Network Architecture level* FIC returns a referral to the URI level FIC where the mapping is stored (message 3 in Figure 3.7) and, subsequently, the local FIC requests the resolution towards it (message 4 in Figure 3.7). Finally, the *URI level* FIC replies with the original URI (message 5 in Figure 3.7), which is then delivered to the FIXP (message 6 in Figure 3.7).

For requesting the mapping of a given URI on a specific network architecture, the FIXP can request its resolution using the same procedure as described above but, in this case, also identifying the target architecture.

3.3.3 FIFu control plane protocol

To support the control procedures between the entities that compose the presented FIFu framework, a control protocol, named the FIFu protocol, was defined. It enables the communication between (a) FIC and FIXPs; and (b) FICs; providing a set of interfaces (Figure 3.8) for that purpose.

The purpose of this protocol is to allow FICs to manage the underlying FIXP instances, including the monitoring and configuration of several operational parameters, the connections between FIXPs, as well as to allow the FIXPs to resolve foreign URIs into their original form or vice versa. Table 3.1 provides a brief description of the messages belonging to each interface. Notwithstanding, this list is not closed and may be enhanced with new messages to fit future purposes.

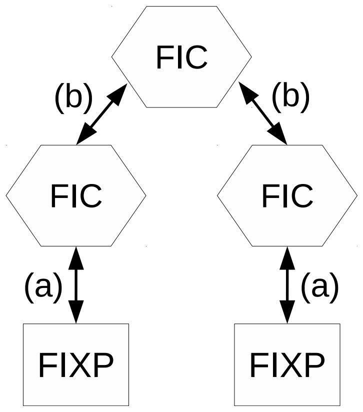


Figure 3.8: FIFu protocol communication interfaces

Table 3.1: FIFu protocol messages

(a) between FICs and FIXPs. Used by	
Get_Operation_Param	FICs to request the underlying FIXPs several operational parameters. The operation parameters include their capabilities (e.g., hardware specs and usage), supported architecture protocol endpoints and flow statistics.
Manage_Endpoints	FICs to add, update or remove architecture protocol endpoints from its underlying FIXPs.
Manage_Events	FICs to (un)subscribe events related with changes of one or more operational parameters. These subscriptions can be configured to operate based on thresholds that generate events whenever crossed, or to periodically trigger the event at a configured time interval.
Manage_Routes	FICs to (re)configure the routes between the FIXPs under its domain (including routes to FIXPs outside its domain).
Manage_Mappings	FICs to install/modify/delete new/existing mappings between original and foreign URIs in the underlying FIXPs.
Resolve_Uri	FIXPs to request the resolution of one or more URIs into their original form or to URIs of a protocol supported by a specific architecture.
Push_Event	FIXPs to report changes in operational parameters whenever they cross a previously specified threshold level. It can also be generated at specified intervals.
(b) between FICs. Used by	
Resolve_Uri	FICs to request the resolution of one or more URIs into their original form or to URIs of a protocol supported by a specific architecture.
Exchange_FIXP_Info	FICs to exchange information about their underlying FIXPs with each other, so that they can be aware of network changes in the surroundings the FIXPs.

3.3.4 Generating foreign URIs for resources

The foreign URI of a given resource can be generated either statically (e.g., resource provider registers an URI for a specific resource) or automatically, as FIXPs need to make the resource available under a different architecture. Independently on how foreign URIs are generated, they must meet two main properties: (i) request for the URI is forwarded towards a FIXP; and (ii) hold information to uniquely identify the resource on the original network architecture.

3.3.4.1 Inferred identifiers

If the address in the original network can be inferred from the received message (for example as seen in IPv4-Mapped IPv6 Addresses - RFC 4291[1]) and if both architectures can share the layers above the network layer, then the mapping registration in the FIC can be skipped. The FIC is able to configure the FIXP to detect these occurrences and to independently translate the URI and, consequently, convert the received message.

3.3.4.2 Auto-generated identifiers

In the remaining cases, upon the reception of requests to translate an URI into its correspondent in a foreign network architecture, if the mapping does not exist, the FIC automatically generates an entry not only for that network architecture but also for other supported foreign network architectures. This request can be done e.g., by the FIXP while adapting message contents.

However, during the generation of identifiers, no information regarding the resource itself is owned, except the URI on the original network architecture. In this way, if the resource is immutable and self-certified by its identifier on the original network architecture, on foreign network architectures the same resource must be considered and treated as being a mutable resource. Consequently, the resource in the foreign network architecture is not addressed by a self-certified identifier (this issue is discussed in Section 3.6). Complementary security mechanisms, supported by the foreign network architecture, may be required in order to achieve a security level (e.g., integrity and authenticity) for a given resource compliant to that offered by its original network.

3.3.4.3 Static identifiers registration

The previous strategies allow resources to be made available in different network architectures in a transparent way for the resource provider. Nevertheless, the resource provider can still register specific foreign identifiers for its resources if desired.

For that, the resource provider must register the foreign identifiers of the resource in the FIC infrastructure. Different strategies can be used:

- individual registration of each resource;
- aggregate a set of resources (using wildcard characters) where only the common part is converted;
- define a conversion algorithm to be applied to (part of) the original URI of the resource;

[1]RFC 4291 - https://www.ietf.org/rfc/rfc4291.txt

- any combination of the previous.

3.3.5 Example of mapping URIs between architectures

In Figure 3.9, a mapping example between different network architectures is presented. This example depicts the mapping from HTTP (over IP) to NDN and PURSUIT network architectures.

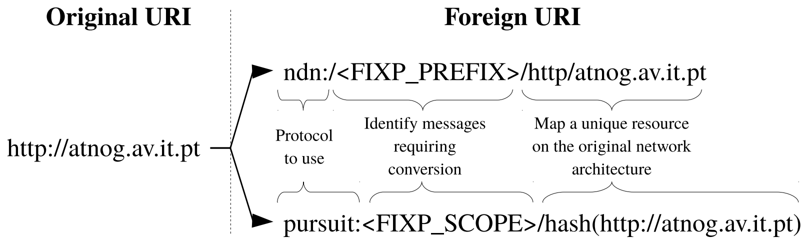

Original URI
Foreign URI
http://atnog.av.it.pt
ndn:/<FIXP_PREFIX>/http/atnog.av.it.pt
Protocol to use
Identify messages requiring conversion
Map a unique resource on the original network architecture
pursuit:<FIXP_SCOPE>/hash(http://atnog.av.it.pt)

Figure 3.9: Example of mappings from HTTP (over IP) to NDN and PURSUIT

Foreign URIs can be divided into three parts: (i) the protocol to use, which is identified in the scheme part of the URI; (ii) a prefix, which is used in the foreign network architectures to identify messages requiring conversion and, therefore, being forwarded towards a FIXP; and (iii) a part that maps a unique resource on the original network architecture, which is used to lookup for the original URI of the resource.

In the NDN case, the foreign URI is generated using a simple concatenation of the original URI, with a prefix associated with the FIXP infrastructure and the scheme of the protocol to use. In PURSUIT, after the scheme part, the foreign URI is composed by the scope associated with the FIXP infrastructure and the remaining part of the URI consists on a hash of the original URI. However, these are just examples on how the foreign URIs could be generated, with the proposed framework being flexible to accommodate different algorithms for foreign URI generation.

3.3.6 Discovering existing resource identifiers

When a new network architecture is connected to the FIXP or when a new resource is deployed in an existing architecture, resource discovery is a fundamental step for a correct operation of the presented framework. In this way, a simplified discovery mechanism was developed for FIFu, in order to allow the framework to be validated. However, such discovery procedure, in its complete form, should be further enhanced. Despite that such enhancement is outside the scope of this thesis, an initial set of possibilities is presented.

As described in Section 3.3.4.3, FICs can discover new resources with the registration of foreign URIs by the resource providers. However, a minimal set of URIs will be discovered this way, since it is not efficiently manageable.

The resources themselves are another source of existing resource identifiers, mainly web resources (such as, HTML pages). In doing so, during content conversion by the FIXP to be compatible with the destination architecture (as described in Section 3.3.1), the FIC can discover new resources whenever the FIXP tries to resolve unknown original URIs into their correspondent in the destination architecture.

Finally, since search engines are widely used for discovering resources in the Internet, they may play an important role in the discovery of resources during the bootstrap of new network architectures and deployment of new resources on an existing network architecture. Also, specialized search engines targeting resources of different network architectures may communicate with each other to exchange identifiers which are subsequently resolved with the FIC.

3.3.7 Deployment considerations

The presented framework aims at enabling a transparent interoperability procedure between different network architectures and, therefore, the operation of existing mechanisms, entities and networks should not be affected by its deployment.

It is envisioned that the presented FIXP could be coupled with the existing Internet Exchange Point (IXP) [8] infrastructure, which consists on network points for exchanging traffic between networks from different Internet service providers (i.e., autonomous systems). Traffic not requiring conversion would flow normally as it happens in today's Internet, while the remaining traffic (i.e., traffic requiring conversion) would be redirected to the FIXP for further processing and conversion. Traffic requiring conversion would be identified by well-known addresses on each network architecture. For example, in current IPv4/IPv6 networks (a set of) IP address(es) and a hostname could be used to identify and route IP messages towards the FIXP, while in ICN approaches, such as NDN and PURSUIT architectures, a reserved prefix or scope ID, respectively, could be (globally) defined and used to forward the ICN messages towards a FIXP.

Moreover, to cope with the high amount of traffic requiring conversion, the FIXP can be considered as a network function capable of being instantiated and virtualized on demand over orchestrated elastic network resources, so that new FIXPs instances can be launched and terminated dynamically whenever needed to properly face the traffic demand. Since the FIXP holds no state outside of the mappings it configures and the ongoing requests and responses, both vertical and horizontal scalability strategies are possible to be applied.

Since the traffic requiring conversion needs to go through the FIXP before being redirected towards its final destination, the FIXP is seen as an anchor point for messages requiring conversion. As such, the deployment of the FIXPs near to the network edge (e.g., exploiting Edge Computing [45] concepts) could alleviate the anchoring-related issues (such as, longer and/or suboptimal forwarding paths).

Finally, with the upcoming 5G networks introducing the concepts of network slicing, different network architectures could be instantiated in different network slices (e.g., an ICN network slice [89]). As such, the FIXP can be connected to slices of different network architectures, enabling it to communicate on each network architecture. In such scenario, SDN technologies may play an important role at redirecting network traffic towards the FIXP, such as when traffic traversing the network belongs to a different and unknown network architecture.

3.3.7.1 Adding new architectures/protocols

The flexible nature of the presented framework makes the addition of the support for new architectures/protocols a matter of network management. More specifically, the FIFu infrastructure operator must provide:

- **the architecture protocol endpoint instance**, so it can then be deployed by FICs in the underlying FIXPs that need to support the new protocol/architecture. With this, FIXPs can understand the new protocol/architecture, being able to receive, identify, process, create and send messages of that protocol/architecture. By understanding the new protocol, knowing how to create a message from an URI and how to apply a set of metadata into the protocol specifics, the FIXP can convert messages between protocols of different network architectures without requiring conversion algorithms on a case by case basis (i.e., for each pair of protocols). The architecture protocol endpoint instance is responsible for deciding which messages require translation, namely if they correspond to the transferral of information or if they are related with control procedures of the corresponding protocol/architecture.

- **information about the URI composition of the protocol/architecture**, allowing the FICs to create correct mappings for that protocol/architecture. In practice, new FIC elements, corresponding to the *Network Architecture level* FIC (Figure 3.6) that understand how these mappings are done, are instantiated.

3.3.8 Answer to challenges

The main features of the presented framework are shown in Table 3.2, along with a brief description on how these features tackles the interoperability challenges identified in Section 3.2.2.

Table 3.2: Solving interoperability challenges through FIFu framework

Feature	**Description**	**Solution to**
Gateway-based approach	Due to the gateway-based approach followed by the presented framework, end-to-end connections are split into two different connections (one on each network architecture), with the FIXP behaving as the intermediary on the communication between both endpoints. Network messages are sent by applications using known protocols and mechanisms towards the FIXP, which converts messages to be compatible with the operational principles of the destination architecture.	Challenge 1
	During the interoperability process, the FIXP acts on behalf of both communication endpoints on each network architecture, emulating the communication paradigms inherent to each architecture.	Challenge 2
Identification of resources using URIs	By using URIs to identify a resource on a given network architecture, the presented framework is able to unify the addresses scattered among multiple layer protocols in a single address (i.e., the URI). In other words, the URI provides all the required information (or hint(s) that lead to the required information) to access the correspondent resource, including the protocol and respective address to use (along with lower layer addresses if required). Therefore, mapping resources using their URIs on each architecture enables an abstraction of address scattering.	Challenge 3
URI resolution mechanism	URIs on the foreign network architecture need to comply with two main properties: (i) enable requests for the URI to be forwarded towards a FIXP; (ii) hold information to uniquely identify the resource on its original network architecture. In the presented framework, the FIC, acting as a resolution system, is responsible for storing and managing the mappings between URIs of different architectures. Thus, the foreign address of the resource is agnostic to the encoding and meaning of its correspondent on the original architecture.	Challenges 4 and 5
Intelligent and adaptation layer separation	Inspired by SDN concepts, the presented framework consists in a two-layer framework: the adaptation and the intelligence layers. The adaptation layer, composed by the FIXPs, is responsible for adapting messages between protocols of different network architectures. In order to do so, the intelligence layer, composed by the FICs, deploys in the FIXPs the knowledge related with the packet format of each protocols (i.e., fields and their semantic meaning), as well as the logic to handle protocol messages and to apply to the messages metadata according the protocol specifics, by means of architecture protocol endpoints.	Challenges 3 and 6
Multiple identifiers for the same resource	A given resource is identified by a single original URI and multiple foreign URIs. While the original URI defines how the correspondent resource is accessed in the network architecture where it is deployed in, the foreign URI defines how the access to the same resource is performed from a foreign network architectures, allowing entities to see the resource as being part of its own network architecture. Thus, applications, mechanisms and entities belonging to a given architecture do not need to be modified in order to access resources deployed in other architectures. The FIXP is then responsible for handling and converting messages between architectures.	Challenge 7

3.4 Simulation results

Preliminary results of the proposed framework in a simulation environment are presented in this section, focusing on a web browsing use case. This evaluation aims not only to demonstrate the practicability of the proposed framework in interoperating different network architectures, but also to provide a first assessment of the cost introduced by such operation, so that it can serve as the baseline for improvements in future research and for the proof-of-concept prototype implementation.

3.4.1 Use case scenarios

To demonstrate the flexibility of accommodating different network architectures (considering their public specifications at the time of writing of this work) while being able to provide interoperability between them, the FIFu framework was exercised to a web browsing use case involving different network architectures. More specifically, the use case scenario includes the interoperability between HTTP over TCP/IP (referred to as IP[HTTP]) and NDN architectures which are oriented on different design paradigms, as well as between PURSUIT and NDN architectures which, although share the same design paradigm, operate based on different communication models.

In both cases, the scenario consists on fetching a HTML page (represented in Source Code 1) deployed in NDN architecture, including other resources that compose the web page.

Source Code 1: HTML Source Code

```
<html>
  <body>
    <img src="logo.png">
    <p>Hello World!</p>
    <a href="ndn:/atnog.av.it.pt">link</a>
  <body>
<html>
```

3.4.1.1 Use case 1: Interoperability between IP[HTTP] and NDN network architectures

The signaling supporting the interoperability scenario between the IP[HTTP] and NDN architectures is depicted in Figure 3.10. HTTP consumers deployed in an IP networking environment can transparently access resources deployed in the NDN architecture.

The HTTP consumer already knows the URI that should be used to access the web page from the IP architecture. It starts by performing a DNS query whose response points to the address of a FIXP, seen in the IP architecture as the *real* provider of

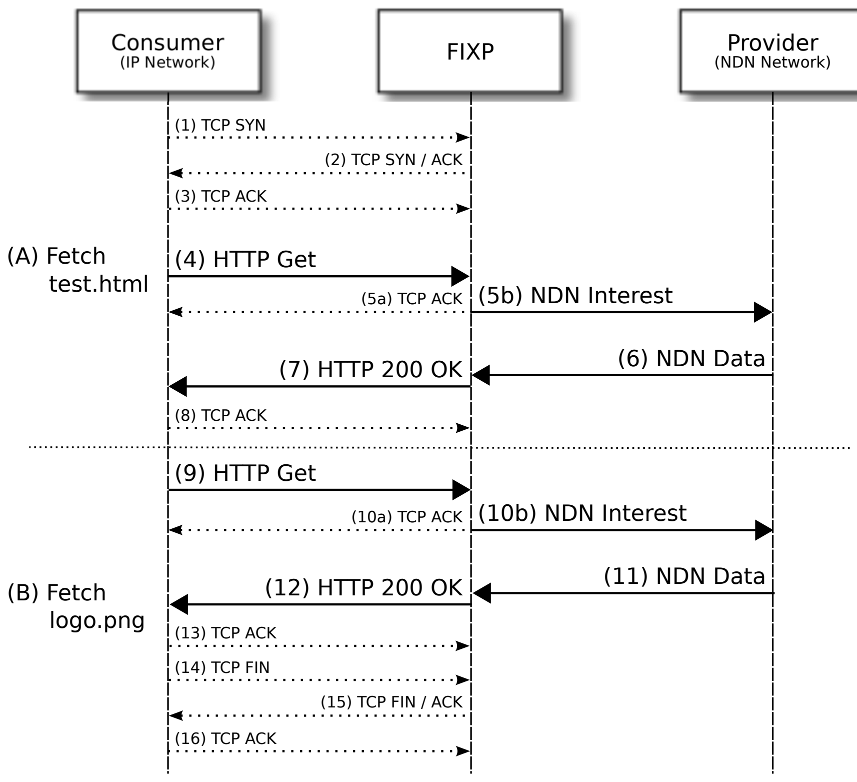


Figure 3.10: Interoperability signaling between IP[HTTP] and NDN

the web page. Since the FIXP is acting on behalf of the NDN producer, the HTTP consumer establishes a TCP connection with the FIXP (messages 1 to 3, in Figure 3.10), after which it sends an *HTTP Get* message requesting the desired resource (message 4 in Figure 3.10). Because TCP control messages are related with the session establishment between the consumer and the FIXP, they are not being converted towards the NDN network, being handled internally by the FIXP. In turn, the *HTTP Get* message is identified as belonging to the transferal of information (i.e., it is identified as the actual request for content) and, therefore, it is converted into an *NDN Interest* message to be sent through the NDN network (message 5b in Figure 3.10). Next, the *NDN Interest* message is routed towards the NDN producer based on the existent mechanisms on the NDN network architecture, without any awareness that the original requester is on a different network architecture. As such, the provider or any entity in the path containing a cached copy of the requested resource reply with a *NDN Data message* towards the FIXP (message 6 in Figure 3.10), benefiting from the existing procedures of an NDN network. Upon reception of the *NDN Data message*, the FIXP extracts the content from the received message which is then used to create the HTTP response message (message 7 in Figure 3.10). In the process, the FIXP inspects the content (in the case of an HTML file) looking for identifiers of other resources (i.e., existing URIs of other resources), replacing them in the content itself by URIs compatible with the IP network architecture. If the mappings are not known, the FIXP can request the FIC for the URIs to be used in the IP architecture. In this way, the FIFu framework is able to maintain a full compatibility of resources in the foreign network architectures, allowing the conversion procedure to be transparent for both communication endpoints (i.e., the consumer and the resource provider). Other resources that compose the HTML page are acquired using the same procedure described above (messages 9 to 13 in Figure 3.10), after which the TCP connection between the consumer and the FIXP is terminated (messages 14 to 16 in Figure 3.10).

3.4.1.2 Use case 2: Interoperability between PURSUIT and NDN architectures

The signaling exchanged between the involved entities when considering the interoperability scenario between PURSUIT and NDN architectures is presented in Figure 3.11. In this evaluation, the PURSUIT URIs are generated without previous knowledge of the content itself and, therefore, the generated URIs are not self-certified (see Section 3.6).

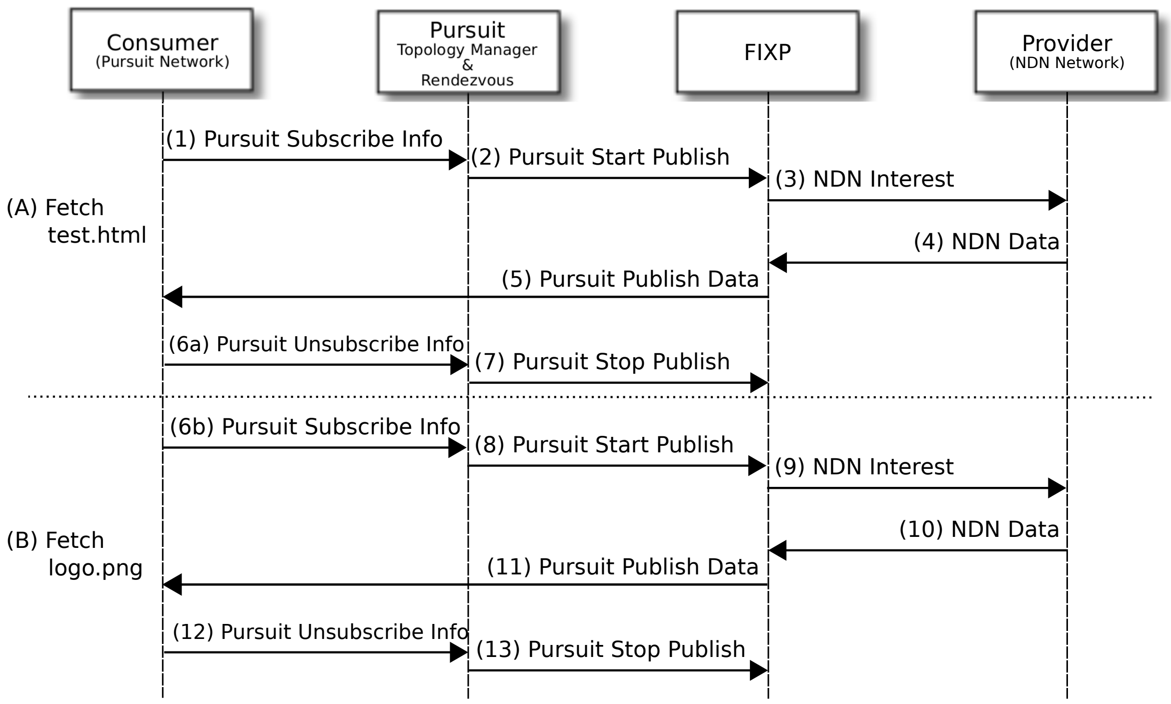


Figure 3.11: Interoperability signaling between PURSUIT and NDN

Upon the discovery of new resources, the FIXP advertises them in the *Rendezvous Node* on behalf of the NDN provider, being seen in the PURSUIT network as the *real* provider. For requesting the web page, the PURSUIT consumer starts by subscribing it towards the *Rendezvous Node* (message 1 in Figure 3.11). The *Rendezvous Node* requests the *Topology Manager* to calculate the forwarding identifier between the FIXP and the PURSUIT consumer, sending it to the content publisher on the PURSUIT network (i.e., the FIXP) (message 2 in Figure 3.11). The subscription, being identified as a transferal of information, is converted by the FIXP into a *NDN Interest* message and sent to the NDN network (message 3 in Figure 3.11) according to the architecture protocol endpoint configured at the FIXP. As in the previous example, entities in the NDN network are not aware that a conversion procedure occurred and, therefore, messages are routed towards the original content provider. The original content provider, or any entity containing a cached copy of the requested content, replies with a *NDN Data* message (message 4 in Figure 3.11). Upon reception, the FIXP extracts the content from the *NDN Data* message, converting existing identifiers to match the PURSUIT architecture. In the process, if the FIXP discovers yet unknown resources, it requests the FIC for the URIs to be used in the PURSUIT architecture, advertising itself in the *Rendezvous Node* as the provider of the referred resources. The content is then encapsulated in a *PURSUIT Publish Data* message (message 5 in Figure 3.11) to be forwarded towards the original requester. Upon the reception of the HTML content, the client requests the unsubscription with the *Rendezvous Node* (message 6a in Figure 3.11), which is then propagated to the FIXP (message 7 in Figure 3.11). The PURSUIT consumer requests subsequent resources belonging to the web page using the same procedure described above (messages 6b and 8 to 13 in Figure 3.11).

3.4.2 Performance evaluation

The evaluation scenario (Figure 3.12) is composed by a set of clients (C1, C2 and C3), each supporting a single network architecture stack (NDN, IP and PURSUIT respectively), and two NDN content providers (P1 and P2) supporting only NDN procedures. The content provider P1 is connected to the FIXP via Router 2 which supports only the NDN stack, while the remaining entities are connected to the FIXP via Router 1 which supports the IP, NDN and PURSUIT stacks. The FIXP is coupled with a simplified set of FIC functionalities to support URI generation and storage.

This scenario was implemented in a simulation environment using ns-3[2], enhanced with NDN (ndnSIM 2.0 [7, 74]) and PURSUIT (Blackadder[3]) network stacks. The simulation environment runs on a virtual machine (with two 3.33GHz CPU cores and 2GB of RAM) hosted in an OpenStack Platform.

Using the proposed framework, besides the NDN client, both IP[HTTP] and PURSUIT clients are able to acquire the web page (i.e., the HTML and PNG files with size 119 and 6132 bytes respectively) deployed in the NDN producers (i.e., *P1*

[2]ns-3 - https://www.nsnam.org/

[3]Blackadder - http://www.fp7-pursuit.eu/

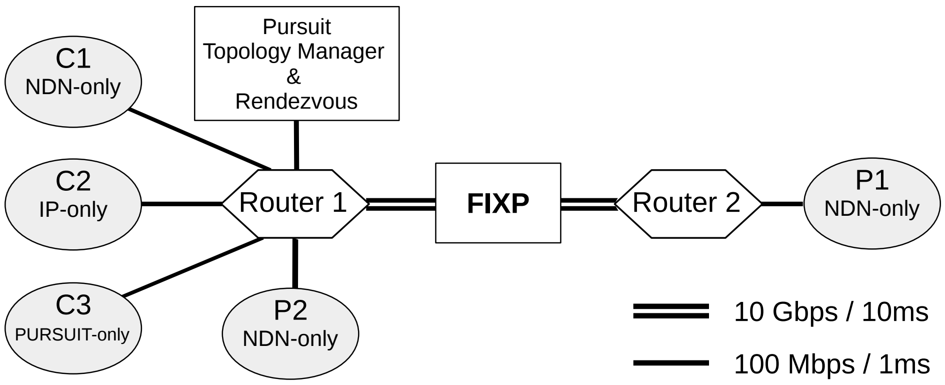


Figure 3.12: Evaluation Scenario

and *P2*), in a transparent way to both the end-user devices and network entities on each network architecture.

In Table 3.3, the signaling size on each network architecture (for a single link) resulting from the interoperability process is presented, with the impact on the total exchanged information when fetching the web page from *P1* and *P2* being shown in Figure 3.13. The time required to fetch the web page from each network architecture when content is deployed in *P1* or *P2* is presented in Figure 3.14. To better evaluate the impact of the proposed framework in enabling the interoperability between different network architectures, the results while fetching the web page from a NDN consumer are also presented for comparison purposes (i.e., in a scenario where no interoperability mechanisms were needed). A scenario (referred to as "IP reference") where all entities are IP-enabled (including both providers *P1* and *P2*) was also evaluated, so it could provide reference values of the current Internet approach. Finally, Figure 3.15 details the delay introduced by the FIXP due to message conversion.

Table 3.3: Signaling size regarding each network architecture (for a single link)

	Total Size w/ content (bytes)	**Signaling size wo/ content (bytes)**	**Exchanged messages**
C1 NDN	NDN: 7078	NDN: 827	NDN: 4
C2 IP[HTTP]	IP: 7345 NDN: 7078	IP: 1093 NDN: 827	IP: 19 NDN: 4
C3 PURSUIT	PURSUIT: 9355 NDN: 7078	PURSUIT: 2929 NDN: 827	PURSUIT: 10 NDN: 4
IP Reference	IP: 7345	IP: 1093	IP: 19

In terms of the amount of signaling exchanged in the foreign network architecture, fetching the web page from the IP[HTTP] and PURSUIT clients, when compared with a native NDN approach, required an higher amount of exchanged messages.

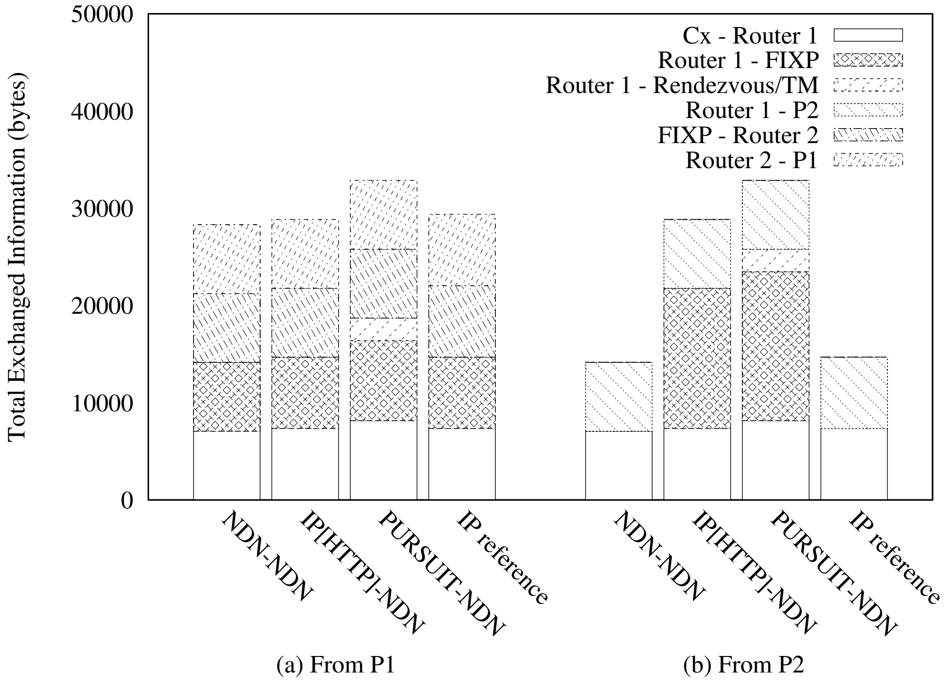


Figure 3.13: Total exchanged information

This changed from 4 messages in NDN to 10 and 19 messages in PURSUIT and IP[HTTP], respectively. If in the IP[HTTP] approach such increase was mainly due to the TCP control signaling as well as the configured IP MTU (due to which 5 TCP Data messages were required to transfer the whole PNG file), in the PURSUIT approach the communication with *Rendezvous Node* required to (un)subscribe each resource was responsible for the increase of exchanged messages. This was also reflected on the total size of the exchanged messages, representing an increase of 3.8% and 32.2% for the IP[HTTP] and PURSUIT approaches, respectively. Considering just the overhead introduced by the messages, the protocols in the IP and PURSUIT architectures exchanged, respectively, more 166 and 2102 bytes. In the original network architecture, the same NDN messages were exchanged independently of the requester and, therefore, the number of messages and their size remained unchanged.

If contents are retrieved from provider *P1* (Figure 3.14a), IP[HTTP] and PURSUIT clients required more time to fetch the web page when compared with a native NDN approach, representing an increase of 99% (roughly double) and 17% respectively. The main reason for such behavior is the delay introduced by the internal mechanisms of both the IP[HTTP] and PURSUIT architectures. In the IP[HTTP] use case, if this increase was due to the TCP control signaling and the configured IP MTU (due to which 5 TCP Data messages were required to fetch the PNG file), in the PURSUIT use case it was due to the communication with the *Rendezvous Node.* Even so, all the previous approaches had lower fetching times in the evaluation scenario when compared

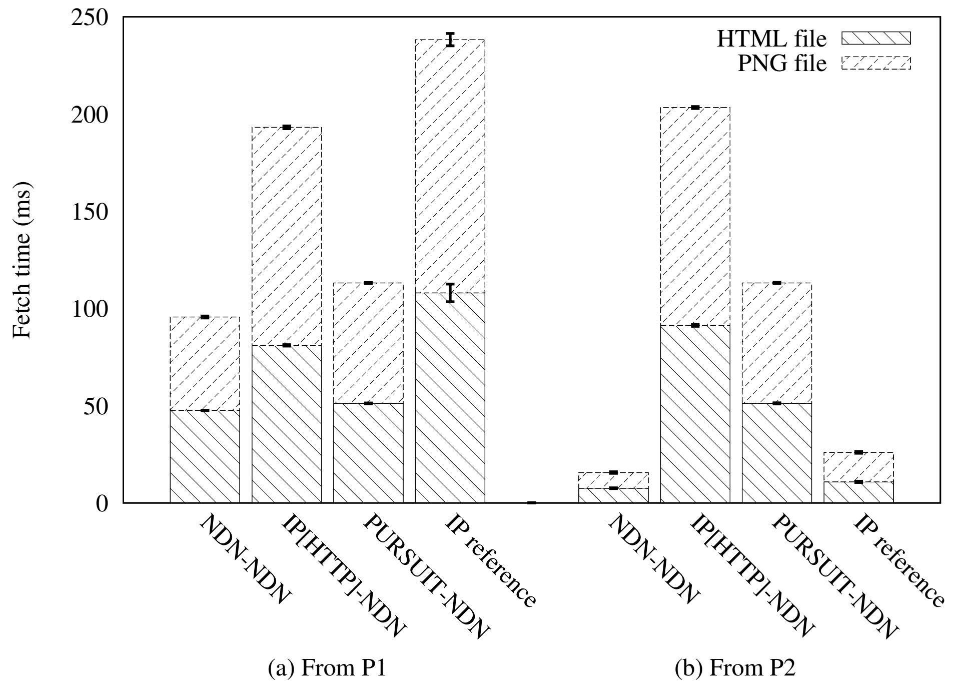


Figure 3.14: Fetching time

to the IP reference. This behavior is due to the delay introduced by the TCP control signaling raised because of the higher RTT between the TCP endpoints in the IP reference approach. Regarding the total exchanged information (Figure 3.13a) the higher amount of exchanged information to fetch the web page from IP[HTTP] and PURSUIT clients, when compared with a native NDN approach, is mainly due to the signaling overhead introduced by both HTTP (over TCP/IP) and PURSUIT protocols.

When considering a nearer provider *P2* and comparing with a native NDN approach, the gap in terms of fetching time (Figure 3.14b) and total exchanged information (Figure 3.13b) while acquiring the web page from IP[HTTP] and PURSUIT clients is even higher. This is due to the anchoring-related issues (i.e., longer and/or sub-optimal forwarding paths) introduced the proposed framework. While the messages in the NDN-only approach only go through *Router1* between the client and the provider *P2* (i.e., *C1↔Router1↔P2*), in IP[HTTP] and PURSUIT approaches messages need to be forwarded to the FIXP in order to be converted to compatible messages in the destination network architecture (i.e., *C2/C3↔Router1↔FIXP↔Router1↔P2*).

The delay introduced by the FIXP while converting messages from one architecture to the other (Figure 3.15) is approximately 1.3% for IP[HTTP] and 3% for PURSUIT of the total communication time. Converting IP[HTTP] or PURSUIT request messages to be compatible with the NDN architecture (i.e., to *NDN Interest* messages) is less time-consuming than the reverse process (i.e., from an *NDN Data* messages), mainly due to integrity check mechanisms. This is clearer in the IP[HTTP] use case, because

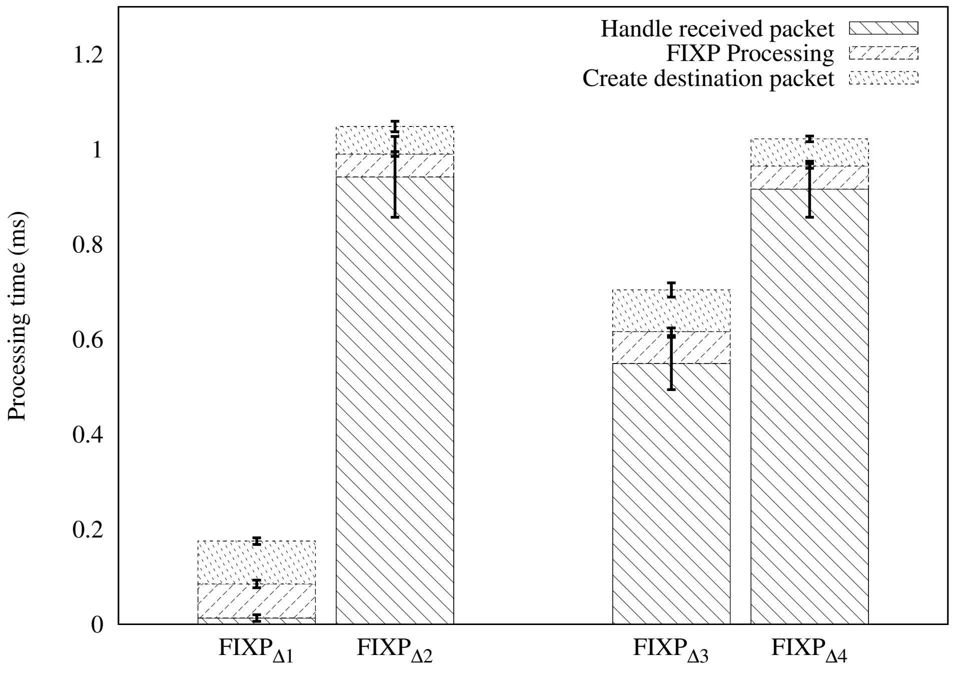


Figure 3.15: Detailed FIXP delay. (FIXP$_{\Delta 1}$: Conversion from IP[HTTP] Request to NDN Interest; FIXP$_{\Delta 2}$: Conversion from NDN Data to IP[HTTP] Response; FIXP$_{\Delta 3}$: Conversion from PURSUIT Start_Publish to NDN Interest; FIXP$_{\Delta 4}$: Conversion from NDN Data to PURSUIT Publish_Data)

the conversion from IP[HTTP] to a *NDN Interest* message is straight forward (i.e., integrity of received IP[HTTP] messages is not verified), while the other way around the integrity of the contents contained in the *NDN Data* message are verified before converting the message to an IP[HTTP] response. This gap is smaller in the PURSUIT case because, in contrast with the previous case, the integrity of the PURSUIT message is also verified before being converted to a *NDN Interest* message.

3.5 Proof-of-concept prototype results

In this section, the implementation results for the proof-of-concept prototype of the proposed framework are presented, focusing on three use cases. This evaluation aims not only to demonstrate the feasibility of the proposed framework in supporting interoperability between different network architectures, but also to provide an assessment on the performance of such operation, serving as baseline for improvements in future research. This evaluation is mainly focused on performing a functionality validation in typical use case scenarios, highlighting the impact introduced by the FIXP and its conversion procedures between network architectures.

3.5.1 Use case scenarios

In this section a set of selected use cases is described, exposing the flexibility and capabilities of the presented framework to enable interoperability between different network architectures.

According to [25], the majority of exchanged traffic in the Internet includes *"Internet video"* and *"Web, email and data"*. As such, the selected use cases will focus in: (i) web browsing; (ii) live video streaming; and (iii) video on-demand. In terms of network architectures, in order to demonstrate the flexibility of the presented framework to provide interoperability between architectures oriented on different designs and/or based on different communication models, the IP, NDN and PURSUIT architectures (considering their public specifications at the time of this writing) were chosen.

For simplifying the signaling description, URI mappings between architectures are assumed to be already known by the FIXP and, therefore, the communication with the FIC is omitted. Nevertheless, the FIXP could use the procedures and signaling described in Section 3.3.2 to discover such mappings. In addition, in all use cases, the user already knows the URI (i.e., the foreign URI) of the resource it wants to access.

3.5.1.1 Use case 1: interoperability in web browsing

Figure 3.16 presents the exchanged signaling that allows NDN-based or PURSUIT-based web browsers to transparently access web pages deployed in today's Internet (i.e., access web pages physically deployed in the IP network).

From a NDN-based web browser: The NDN-based web browser starts by issuing a *NDN Interest* message to express its interest on the desired web page (message 1.1a in Figure 3.16). By receiving this message, the FIXP converts the received *NDN Interest* message into its equivalent in the HTTP protocol to be used in the IP network architecture. As such, the FIXP establishes a TCP connection with the web server, through which it requests (message 2 in Figure 3.16) and receives (message 3 in Figure 3.16) the desired web page via HTTP protocol. Upon the reception of the *HTTP 200 OK* message, the FIXP terminates the TCP connection with the web server. Before sending the content towards the NDN-based web browser, depending on the content type (e.g., HTML page) the FIXP inspects the content in order to find and replace existing identifiers to other resources (i.e., URIs). If mappings for the discovered URIs are not known by the FIXP, it will request the FIC to provide the mappings to be used in the NDN architecture. Thus, contents maintain the compatibility with the foreign network architecture, allowing the conversion procedure to be transparent to

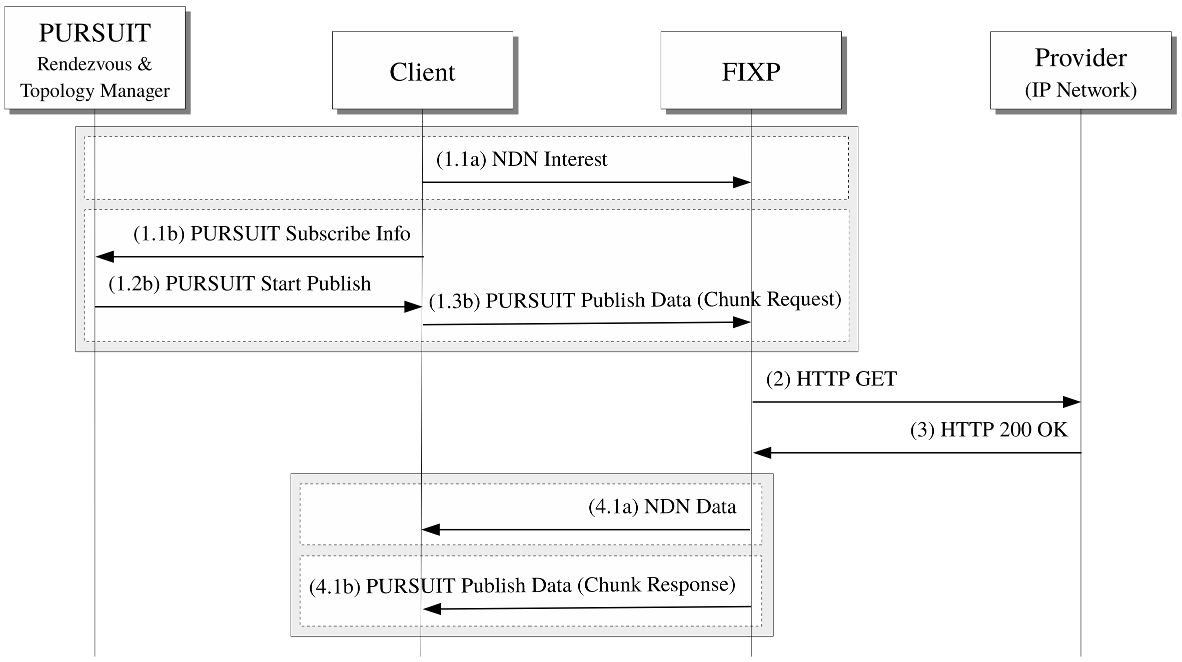

PURSUIT
Rendezvous &
Topology Manager
Client
FIXP
Provider
(IP Network)
(1.1a) NDN Interest
(1.1b) PURSUIT Subscribe Info
(1.2b) PURSUIT Start Publish
(1.3b) PURSUIT Publish Data (Chunk Request)
(2) HTTP GET
(3) HTTP 200 OK
(4.1a) NDN Data
(4.1b) PURSUIT Publish Data (Chunk Response)

Figure 3.16: Signaling for a web browsing use case (NDN/PURSUIT-to-IP)

the endpoints without requiring the intervention of the resource provider. The contents are then sent in an *NDN Data message* (message 4.1a in Figure 3.16) towards the NDN-based web browser. Upon the reception of the HTML content, the NDN-based web browser requests the remaining resources that compose the web page using the same procedure described above.

In this example, a simple communication pattern for web interaction in the NDN network architecture is presented. Nevertheless, the presented framework is flexible to accommodate other (and more complex) communication patterns, as the ones defined e.g. in [78].

From a PURSUIT-based web browser: In this use case, since the FIXP acts on behalf of the web server, it has already advertised the foreign URI associated with the web page at the *PURSUIT Rendezvous Node*, being seen in the PURSUIT network as the real provider.

This use case starts with the PURSUIT-based web browser subscribing the desired web page towards *PURSUIT Rendezvous Node* (message 1.1b in Figure 3.16). The *PURSUIT Rendezvous Node* request the *PURSUIT Topology Manager Node* to compute the forwarding identifiers for both the reverse and forward directions. These are then delivered to the PURSUIT-based web browser in a *PURSUIT Start Publish* message (message 1.2b in Figure 3.16). Upon the reception of this message, the PURSUIT-based web browser starts requesting the web page by sending *Chunk Request* messages encapsulated in *PURSUIT Publish Data* message using the reverse direction identifier to forward the message towards the FIXP (message 1.3b in Figure 3.16). After receiving this message, the FIXP converts the request into an *HTTP GET* message (message 2 in Figure 3.16) so that the web page could be requested in the original network architecture. Upon the reception of the *HTTP 200 Ok* (message 3 in Figure 3.16), the FIXP extracts the payload content from the received response message, as well as any relevant metadata. The content is parsed by the FIXP looking for identifiers of other resources, which are then replaced in the content payload by identifiers compatible with the PURSUIT architecture. The modified content is sent in a *Chunk Response* messages encapsulated in *PURSUIT Publish Data* message, using the forward direction identifier to forward the message towards the PURSUIT-based web browser (message 4.1b in Figure 3.16). Upon the reception of the HTML content, subsequent resources that compose the web page are acquired using the same procedure described above.

3.5.1.2 Use case 2: interoperability in live video streaming

In this use case, a live video streaming is provided through a PURSUIT network, so that the multicast forwarding capabilities of such architecture could be leveraged. The exchanged signaling for video clients on IP and NDN architectures to access the live video stream is presented in Figure 3.17.

From a RTSP-based (over IP) video client: The FIXP, acting on behalf of the video provider, provides the video stream in the IP architecture via RTSP/RTP protocols.

The video client in the IP network starts by fetching a description of the video stream using an *RTSP Describe* of the RTSP protocol (messages 1.1a and 1.2a in Figure 3.17), including streaming media initialization parameters. Using these parameters, the video client establishes an RTSP session with the FIXP (messages 1.3a and 1.4a in Figure 3.17). Since the FIXP knows that the previous messages are related to control procedures regarding the session establishment between the endpoints in the IP architecture (i.e., between the FIXP and the video client in the IP network architecture), these messages are handled internally by the protocol stack, not being converted towards the PURSUIT network architecture. When the video client sends the *RTSP Play* message, it is recognized by the FIXP as the actual request for the video stream (due to its architecture protocol endpoint configuration) and, therefore, the message is converted by the FIXP into a *PURSUIT Subscribe Info* message subscribing the correspondent resource in the original network architecture (messages 2 and 3 in Figure 3.17). While the subscription is active, each piece of the video stream is

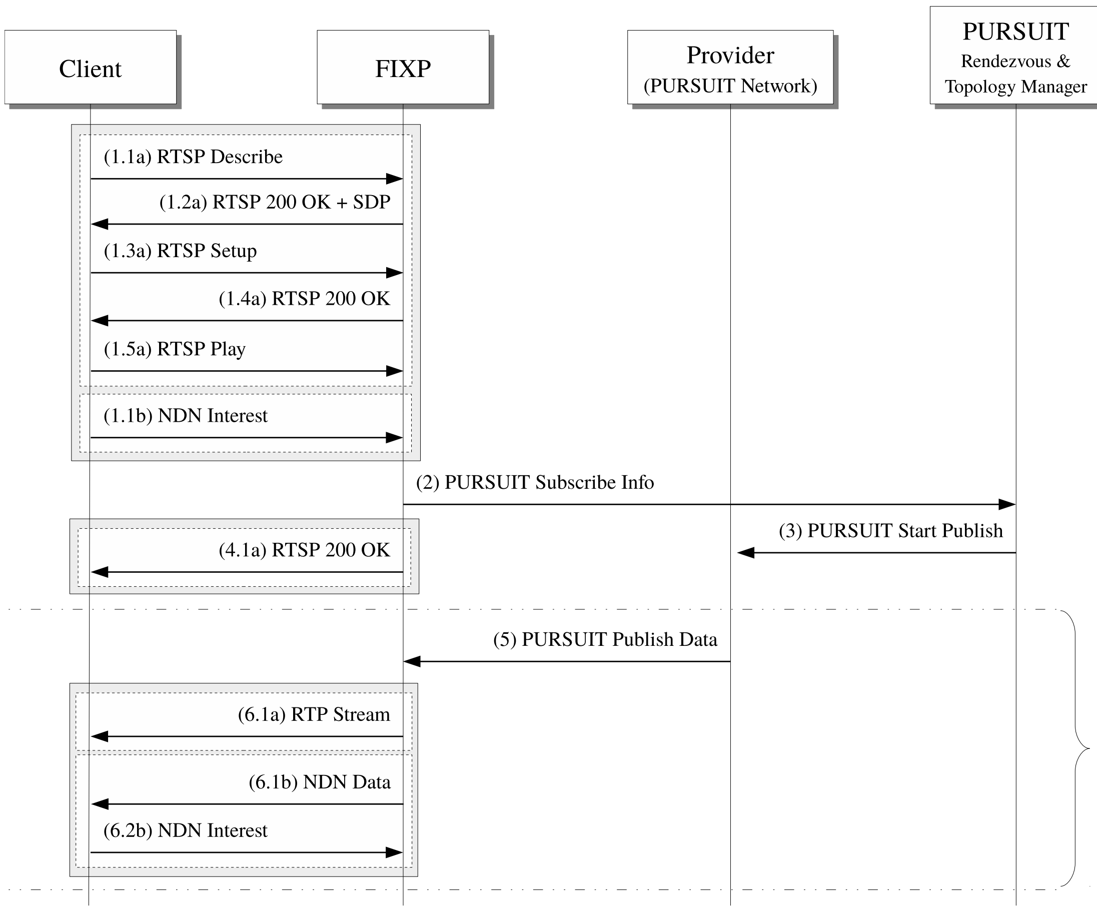


Figure 3.17: Signaling for a live streaming use case (IP/NDN-to-PURSUIT)

proactively sent towards the FIXP through *PURSUIT Publish Data* messages (message 5 in Figure 3.17). The FIXP extracts the content from the received messages, sending it towards the RTSP-based (over IP) video client via the RTP stream (message 6.1a in Figure 3.17) previously negotiated in the RTSP session establishment.

From a NDN-based video client: In order to request the video stream the user wants to have access to, the NDN-based video client starts by issuing a *NDN Interest* message (message 1.1b in Figure 3.17) addressing the desired content. This message is received by the FIXP, which, in turn, triggers the subscription of the video stream in the PURSUIT network architecture as described in the previous case (messages 2 and 3 in Figure 3.17). Consequently, the video provider starts sending the video stream towards the FIXP (message 5). Each received *PURSUIT Publish Data* message is converted into *NDN Data* messages (each corresponding to a different chunk). Due to the request/response approach followed by NDN (i.e., one *Data* message per *Interest* message), for the *NDN Data* messages (message 6.1b in Figure 3.17) being delivered to the NDN-based video client, it needs to continuously poll the next chunks of the video stream (message 6.2b in Figure 3.17). Due to the receiver-driven operation of NDN communication, the NDN-based video client needs to discover or estimate the rate at which it must trigger the *NDN Interest* messages (e.g., as described in [69, 104]).

3.5.1.3 Use case 3: interoperability in video on-demand

After finishing the live video stream, its provider makes the video stream available for on-demand access. For that, it divides the video into several segments and computes the correspondent manifest file containing the metadata for the various video segments that are available. In Figure 3.18, it is presented the exchanged signaling that makes possible for video clients on IP and NDN networks to access the on-demand stream.

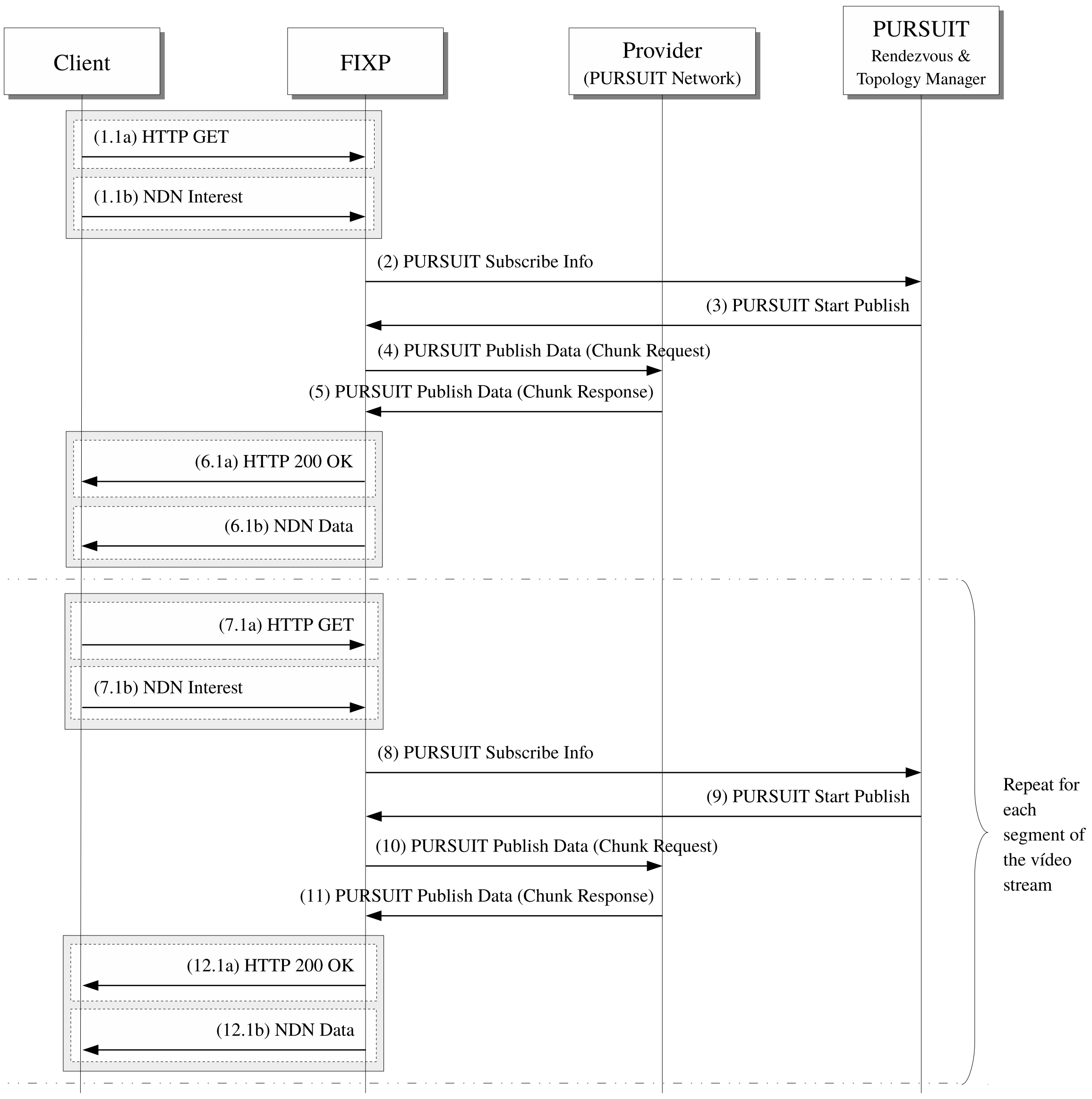

Client
FIXP
Provider
(PURSUIT Network)
PURSUIT
Rendezvous &
Topology Manager
(1.1a) HTTP GET
(1.1b) NDN Interest
(2) PURSUIT Subscribe Info
(3) PURSUIT Start Publish
(4) PURSUIT Publish Data (Chunk Request)
(5) PURSUIT Publish Data (Chunk Response)
(6.1a) HTTP 200 OK
(6.1b) NDN Data
(7.1a) HTTP GET
(7.1b) NDN Interest
(8) PURSUIT Subscribe Info
(9) PURSUIT Start Publish
(10) PURSUIT Publish Data (Chunk Request)
(11) PURSUIT Publish Data (Chunk Response)
(12.1a) HTTP 200 OK
(12.1b) NDN Data
Repeat for each segment of the vídeo stream

Figure 3.18: Signaling for a video on-demand use case (IP/NDN-to-PURSUIT)

From a HTTP-level streaming (HLS) over IP video client: In this scenario, the FIXP makes the video stream available in the IP network architecture via HLS mechanisms. In doing so, the video client in the IP architecture establishes a TCP connection with the FIXP, through which it requests the manifest file, containing the URIs of the several media segments that compose the video stream, via *HTTP GET* message (message 1.1a in Figure 3.18). The TCP control messages are not propagated to the PURSUIT network because they are related with the TCP session establishment between the video client and the FIXP, being handled internally by the FIXP. In turn, the *HTTP GET* message is identified as belonging to the transferal of information, being converted into a *PURSUIT Subscribe Info* message sent towards the *PURSUIT Rendezvous Node* (message 2 in Figure 3.18). The *PURSUIT Rendezvous Node* requests the *PURSUIT Topology Manager Node* to compute the forwarding identifiers for both the reverse and forward directions, which are then delivered to the FIXP in a *PURSUIT Start Publish* message (message 3 in Figure 3.18). Using the reverse direction identifier, the FIXP requests the manifest file by issuing a *Chunk Request* message sent to the video provider via a *PURSUIT Publish Data* message (message 4 in Figure 3.18). The video provider returns the manifest file to the FIXP via *Chunk Response* message encapsulated in a *PURSUIT Publish Data* message (message 5 in Figure 3.18). After receiving the manifest file, the FIXP adapts the content so that URIs contained in the manifest file are compatible with the IP architecture. Lastly, the manifest file is delivered to the video client via HTTP protocol (message 6.1a in Figure 3.18). Each segment of the video stream contained on the manifest file is subsequently requested as needed using a similar procedure to the one describe above (messages 7.1a to 12.1a in Figure 3.18). Finally, the video client terminates the TCP connection with the FIXP.

From a NDN-based video client: As in the IP use case, the NDN-based client initially requests the manifest file of the desired video stream by issuing a *NDN Interest* message (message 1.1b). The FIXP, by receiving this message, expresses its interest on the video stream on the original network architecture by subscribing the manifest file of the video stream (message 2), being retrieved with the forwarding identifiers for both forward and reverse directions (message 3). The FIXP then issues a *Chunk Request* message encapsulated in a *PURSUIT Publish Data* message towards to the video provider using the reverse direction identifier (message 4), to which the video server replies with a *Chunk Response* message sent in a *PURSUIT Publish Data* message towards the FIXP using the forward direction identifier (message 5). The URIs on the manifest file are converted to be compatible with the destination architecture (i.e., the NDN network architecture), which is finally delivered to the NDN-based video client in a *NDN Data* message (message 6.1b). The segments of the video stream are then requested by the video client using a similar procedure to the one described above (messages 7.1b to 12.1b).

3.5.2 Evaluation Scenario

For evaluation purposes, a proof-of-concept prototype of the presented framework was developed, along with a set of example applications for each use case described previously.

The evaluation scenario (presented in Figure 3.19) is composed by the proof-of-concept prototype of the FIXP, coupled with a simplified set of FIC functionalities to support URI generation and storage. Among the remaining entities that compose the evaluation scenario are a set of clients (C1, C2 and C3), each supporting a single network stack (NDN, PURSUIT and IP respectively), the PURSUIT *Rendezvous and Topology* node and a webserver (supporting only the IP architecture) and a video server (supporting only PURSUIT architecture). Both servers are connected to the FIXP via Router 2, and the clients and *Rendezvous and Topology* node are connected to the FIXP via Router 1. NDN and PURSUIT are configured to use IP/UDP as transport protocol between ICN entities.

The network environment of the evaluation scenario supports IP, NDN (NDN Platform v0.4.0[4]) and PURSUIT (Blackadder v0.4[5] enhanced with the mmFTP capabilities [97]) network stacks. Each entity runs on a virtual machine (with two 3.33GHz CPU cores and 2GB of RAM), except the web server that consisted in an existing server in the current Internet (i.e., *www.ua.pt*).

Finally, NDN and PURSUIT chunk size is configured to 4400 bytes. The *Scope* and *Rendezvous* IDs in PURSUIT are setup to 8 bytes and the *forwarding IDs* to 32 bytes, following the default values of *Blackadder v0.4* implementation.

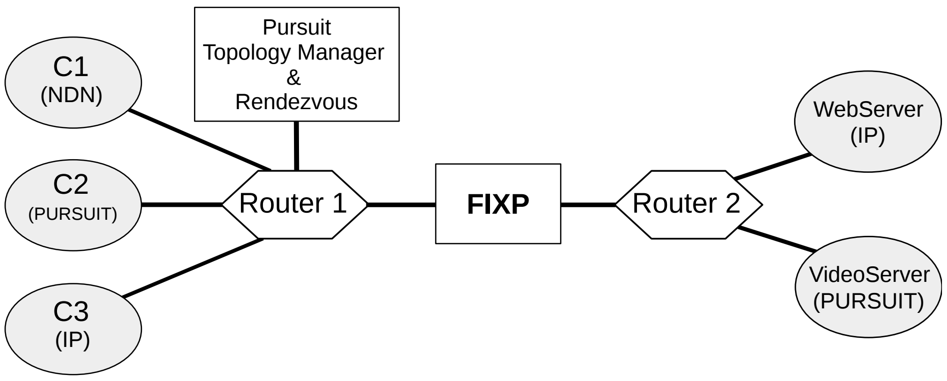


Figure 3.19: Evaluation Scenario

[4]NDN Platform - http://named-data.net
[5]Blackadder - http://www.fp7-pursuit.eu/

3.5.3 Implementation details

Since the initial prototype aimed to assess, evaluate and verify the feasibility of the proposed framework, the FIFu framework was implemented as a standalone application, with the FIXP being coupled with a simplified set of FIC functionalities.

This proof-of-concept prototype is implemented in C++ and it follows a plugin-based approach (as depicted in Figure 3.20), allowing not only new functionalities to be supported without requiring to modify the core of the implementation, but also to be deployed on-the-fly without requiring the restart of the FIXP. Such approach also allows the core of the implementation to be independent of the external plugins. Two different plugins were considered:

- network architecture/protocols plugins (referred previously as the architecture protocol endpoints), which implement the logic related to a given network architecture or protocol. As such, each plugin implements the capabilities of acting as a source and destination of messages regarding a specific network architecture or protocol. Also, each plugin is responsible for generating the foreign URIs whenever requested by the core module, and to advertise them in the correspondent network architecture. For this proof-of-concept prototype, plugins for HTTP, RTSP/RTP, NDN and PURSUIT (with and without support to multipath extensions) were implemented.
- content converters plugins, which handle the conversion of URIs inside specific content types (e.g., HTML). The content converters plugins parse the content looking for URIs inside the content itself and, after requesting the core module for a mapping, replace the original URI by the generated foreign URI. For this proof-of-concept prototype, plugins for HTML and SDP content types were implemented.

Finally, the core module of the FIXP implements the redirection of messages between the network architecture/protocol plugins and, if required, towards the correspondent content converters plugins. In addition, as part of the FIC functionalities, the core module of the FIXP is responsible for requesting the network architecture/protocols plugins for the generation of foreign URIs and for storing the mappings between the original and foreign URIs.

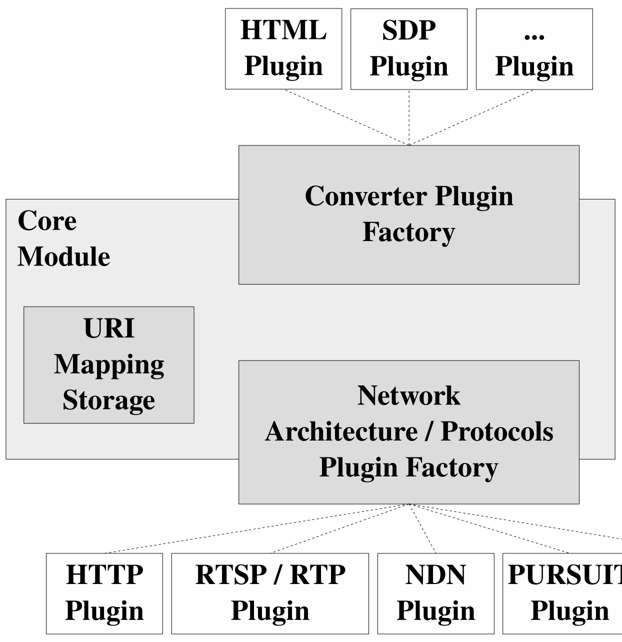


Figure 3.20: Proof-of-concept prototype implementation architecture

3.5.4 Web browsing performance

The evaluation of the proposed framework in the web browsing scenario consisted on fetching an existing web page (i.e., *http://www.ua.pt*[6]) deployed in the IP architecture by NDN and PURSUIT clients. Results are shown in Figure 3.21 and Table 3.4. The former presents the time required to fully load the web page by each client, while the latter presents the signalling size on each network architecture (for a single link). To better assess the impact of the interoperability procedures, values for an IP client are presented as reference values since both client and server supported the same architecture and, therefore, exchanged messages did not required conversion.

From Figure 3.21, it can be observed that when requesting the web page for the first time the delay introduced by the interoperability procedure is higher (approximately 11% for NDN and 17% for PURSUIT) than when compared with subsequent requests. The reason consists on the fact that in the first request the FIXP generates foreign URIs for each resource discovered in the HTML content. In the PURSUIT case, the delay also encompasses the procedures related with the advertising of the identifiers associated with each resource in the *PURSUIT Rendezvous Node*. If the content does not change, the procedure related with the generation of foreign URI is bypassed in the follow up requests for the same content, being handled as *subsequent requests (no cache)*. However, the delay is still higher than in the current IP approach in approximately 55ms and 14ms for NDN and PURSUIT respectively. Otherwise, if the

[6]Frozen as October 10, 2016

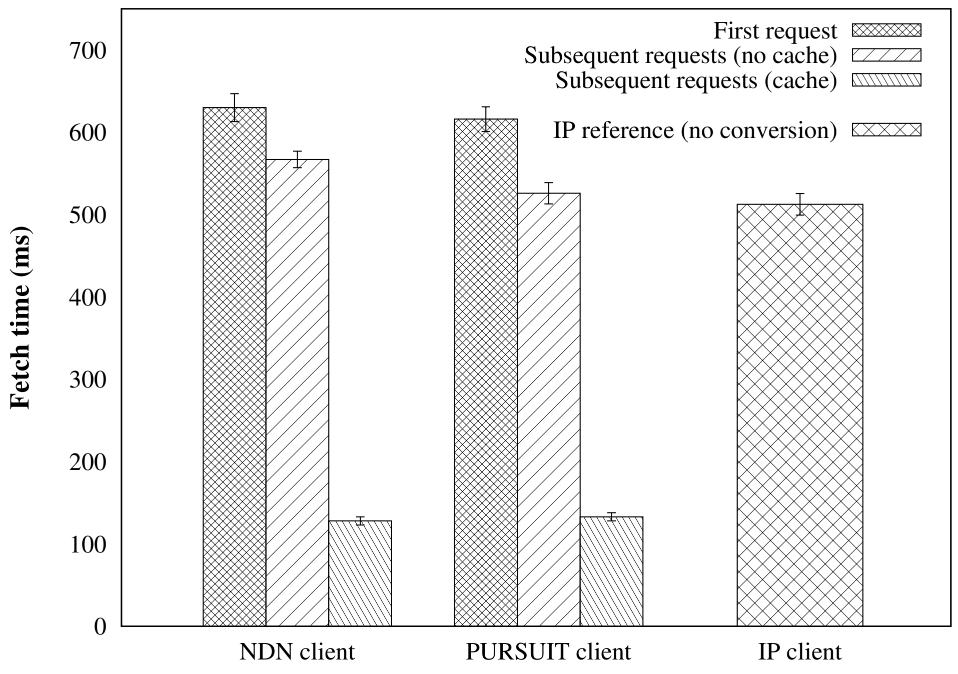


Figure 3.21: Fetching time to fully load the web page in the web browsing use case

content changes, the generation of foreign URIs will only be triggered for unknown resources.

Nevertheless, by enabling the content to be available in the ICN instantiations, in-network caching can be leveraged. In the evaluation scenario, caching capabilities were enabled in *Router 1* of the evaluation scenario. Assuming that the web page and associated resources (e.g., CSS and images) are cached in *Router 1*, clients in the NDN and PURSUIT network architectures were able to load the whole web page, respectively, 4.0 and 3.8 times faster that in the current IP approach. In this case, all resources of the web page were retrieved from the cache in *Router 1* and, therefore, requests were not processed by the FIXP. Note that in-network caching is expected in NDN and PURSUIT architectures.

Table 3.4: Signaling overhead on each network architecture (for a single link)

	Signaling overhead with ICN as overlay over IP/UDP in bytes (%)	**Signaling overhead with ICN over Ethernet in bytes (%)**	**Exchanged messages**
C1 NDN	IP: 60921 ± 363 (17.3%) NDN: 50124 (14.6%)	IP: 60921 ± 363 (17.3%) NDN: 45084 (13.3%)	IP: 728 ± 4 NDN: 180
C2 PURSUIT	IP: 60539 ± 607 (17.2%) PURSUIT: 30574 (9.4 %)	IP: 60539 ± 607 (17.2%) PURSUIT: 23910 (7.5%)	IP: 721 ± 8 PURSUIT: 238
C3 HTTP over IP	IP: 63262 ± 1486 (17.7%)	IP: 63262 ± 1486 (17.7%)	IP: 586 ± 14

In terms of signaling overhead (shown in Table 3.4), an increase of the number of messages in the original network architecture was seen when the request came from NDN and PURSUIT network architectures (and, therefore, requests needed to go through the FIXP) when compared to requests originated in the IP network (and, therefore, not requiring conversion procedures). When interoperability was required, the FIXP created a new TCP connection to fetch each resource of the web page (i.e., no HTTP pipelining) requiring additional TCP control signaling to establish and terminate each connection, while the client in the IP network architecture was able to reuse the same TCP connection to request several resources.

Still, even if more messages were exchanged in the original network when the interoperability occurred, the signalling overhead was smaller. The *HTTP GET* messages issued by the FIXP contained only the minimal required information (i.e., *Host* and *Accept* HTTP headers) to request the desired content, representing approximately less 250-300 bytes per resource request comparing with the IP-only approach. Moreover, the number of exchanged messages and the related signalling overhead in the ICN network architectures was less than the one verified in the IP network architecture.

In Figure 3.22, the delay introduced by the FIXP is detailed for each step of the message conversion for different types of content.

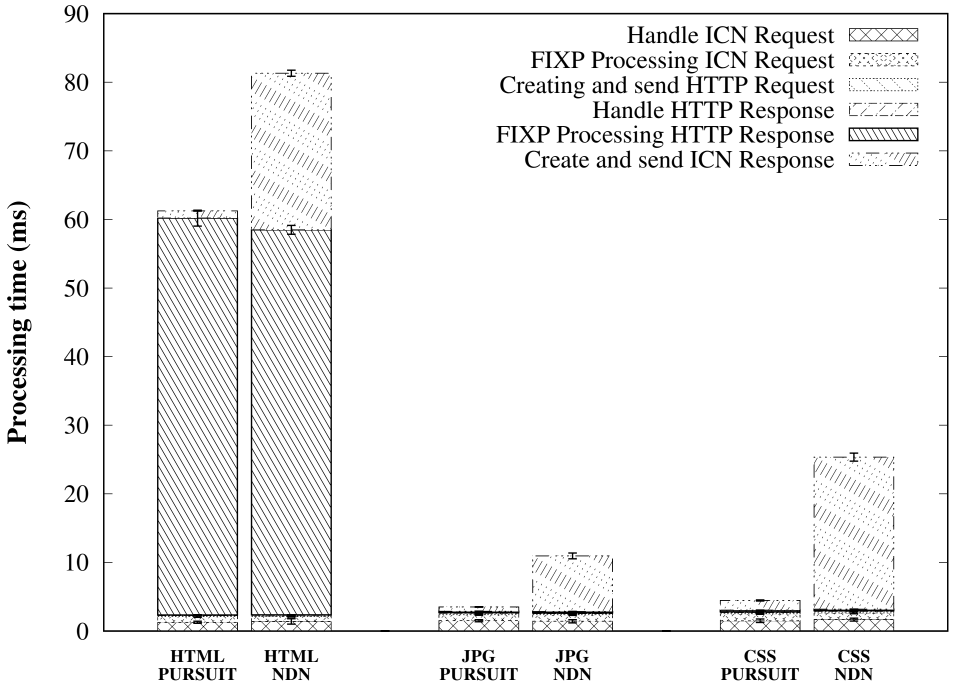


Figure 3.22: Detailed FIXP delay. File sizes: (i) HTML file = 30.7kB; (ii) JPG file = 7.4kB; (iii) CSS file = 30.3kB

With respect to the selected interoperability approaches, handling the conversion of the request message has been found to be less time consuming than the correspondent response message. The reason resides on the fact that converting NDN/PURSUIT requests into a HTTP-compatible request is a straight forward process, consisting on finding the mapping of the foreign URI and creating the correspondent *HTTP GET* message. In turn, the conversion of the HTTP response message into NDN/PURSUIT compatible messages proved to be a more complex process where (i) content is converted if required (e.g. HTML contents) to be in compliance with the destination network architecture (as verified for the HTML content in Figure 3.22, represented in "*FIXP Processing HTTP Response*" step) ; and (ii) content is split in several chunks which are sent in separated messages.

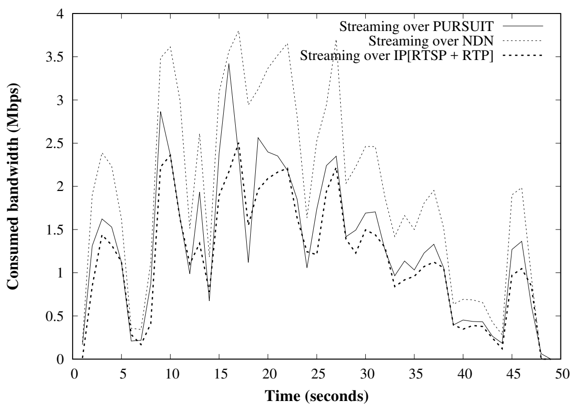
In the case of NDN, since the signature for each *NDN Data* message is calculated and then included in the message itself, the delay associated to the creation and sending of the ICN response messages is higher in NDN than in the PURSUIT case (as verified for all types of content in Figure 3.22, represented in "*Create and send ICN Response*" step). In addition, in the NDN case, the greater the content size, the higher is the number of *NDN Data* messages that need to be sent (since the content is split in more chunks) and, therefore, the higher the time to create and send all the NDN messages (as verified when comparing the "*Create and send ICN Response*" step in Figure 3.22 of NDN for e.g. the JPG and CSS content types of content).

Finally, whenever contents needed to be adapted to be compatible with the destination network architecture, this step (represented in Figure 3.22 as "*FIXP Processing HTTP Response*") demonstrated to be the most time consuming in prototype implementation. Notwithstanding, adapting the contents is not required for all types of content (as observed for the *JPG* and *CSS* files), which significantly reduces the processing time of the response messages in the FIXP.

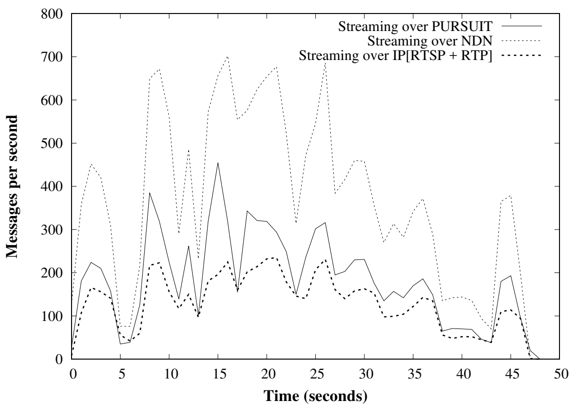
3.5.5 Live video streaming performance

For the live video streaming experiment, a video server deployed in the PURSUIT network provided a live video stream, which was accessed by clients in IP and NDN network architectures. The results of this experiment in terms of consumed bandwidth and number of exchanged messages on each architecture for individual requests by clients on NDN and PURSUIT architectures are presented in Figure 3.23. Performance results regarding the request of the video stream from the PURSUIT client are also presented as reference values to better assess the impact of the remaining approaches.

Regarding the selected interoperability approach, and taking the streaming over PURSUIT values as reference, in Figure 3.23a, it is observed that more bandwidth was required to deliver the video stream via NDN. This increase is due to two main factors:



(a) Used bandwidth



(b) Exchanged messages

Figure 3.23: Performance of live video scenario on each network architecture

(i) in the NDN architecture each chunk of the video stream is explicitly requested via an *NDN Interest* message, contrarily to the PURSUIT architecture where the video is subscribed once, leading to an increase of the number of exchange messages (as can be verified in Figure 3.23b); and (ii) since more metadata is sent in the *NDN Data* messages (e.g., signature related information), the signaling overhead caused by the *NDN Data* messages is larger than the one caused by the *PURSUIT Publish Data* messages.

In what regards the access to the video stream by a client in the IP network, the FIXP had no previous knowledge about the original video parameters during the RTSP session establishment with the video client (this issue is further detailed in Section 3.6.2). As such, in the FIXP implementation a default set of parameters had to be used and, consequently, the FIXP had to re-encode the received video stream to match the parameters agreed during the RTSP session establishment. Aiming for a fair comparison among the different architectures, in this evaluation the stream in the IP architecture replicated the parameters of the original one.

In Figure 3.23a, it is observed that delivering the video stream to the client via the IP architecture required slightly less bandwidth than via the PURSUIT network architecture. Among the reasons are the lower overhead introduced by the RTP message when compared with the *PURSUIT Publish Data* messages and the fact that, during the re-encoding of the video stream, the content from several PURSUIT messages was aggregated in a single RTP message, which reduced the number of exchanged messages as well (as can be observed in Figure 3.23b).

The impact on the bandwidth usage whenever simultaneous requests from different network architectures occurs (i.e., from clients in the IP and NDN architectures) was also measured. In doing so, the video stream was initially requested by the NDN-based video client and after 10 seconds, the client in the IP architecture requested the same video stream. Results are presented in Figure 3.24, where it can be observed that the second request did not affect the bandwidth consumption on the PURSUIT network architecture. The reason resides on the fact that the FIXP caches incoming messages if the original resource is already being requested by a previous message. Afterwards, upon the reception of the requested resources by the FIXP, these are delivered to the multiple pending requesters.

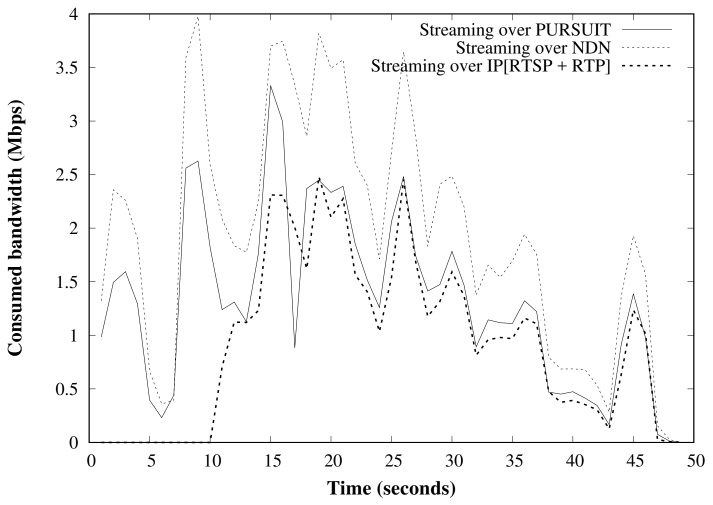


Figure 3.24: Bandwidth on simultaneous requests in live video scenario

3.5.6 Video on-demand performance

In order to make the video captured in the previous scenario available for on-demand access, it was divided into multiple segments and the correspondent manifest file was generated. As such, in this experiment the on-demand video stream continued to be provided through a PURSUIT network architecture and it was, afterwards, accessed by clients in the IP and NDN network architectures. Figure 3.25 shows the results of this experiment, presenting the consumed bandwidth on each network architecture. Like in the previous evaluations, values for the access by a PURSUIT client, which did not require message conversion (i.e., both the client and the server support the same network architecture), are also presented as a reference to better assess the impact of the other approaches.

As can be observed in Figure 3.25, in the video on-demand experiment, the gap between the bandwidth consumption in the NDN and PURSUIT network architectures is not as great as in the live video streaming use case. Two reasons are on the base of this result.

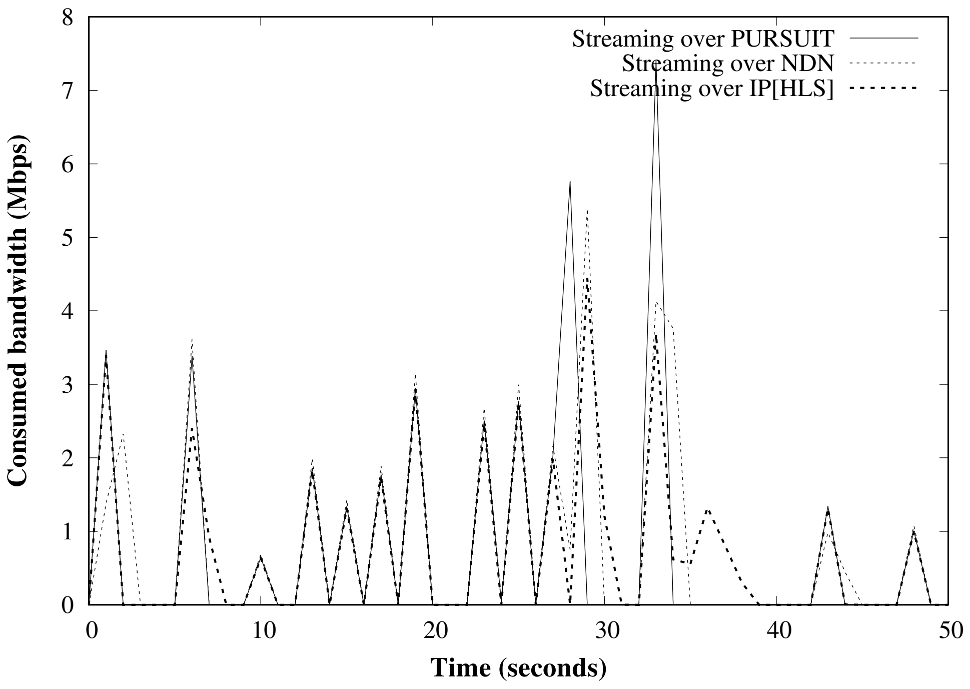


Figure 3.25: Performance of video on demand scenario on each network architecture in terms of consumed bandwidth

First, in this experiment the *NDN Data* messages carried full size chunks, transporting, according to out evaluation setup, 4400 bytes of information. In contrast, in the live video use case, the amount of information transported on the *NDN Data* messages depended on the size of the captured video on each instant. As a result, the protocol overhead is less impactful regarding the overall communication.

Secondly, unlike in the live video use case where the video was subscribed once and the information delivered in several messages, in this experiment each individual chunk of each video segment and manifest files was explicitly requested in the PURSUIT network architecture. As a result, the same number of (i) *NDN Interest* and *PURSUIT Chunk Requests*; and (ii) *NDN Data* and *PURSUIT Chunk Response* were exchanged. In the PURSUIT network architecture, additional messages were exchanged between the video client and the *Rendezvous Node* related with the subscription of the manifest and segments files and retrieval of the forwarding identifiers for both the reverse and forward directions (corresponding to messages 2-3 and 8-9 in Figure 3.18).

When the request was made by the client in the IP network architecture, both the manifest and segment files were delivered via HTTP protocol. Therefore, unlike the previous use case, it was not required to re-encode the video stream. As a result, in terms of consumed bandwidth it was very similar to the one verified in the PURSUIT network architecture.

3.5.7 Generating foreign URIs

Whenever a new resource is discovered by the FIFu framework, URIs to be used (in the network architectures besides the one where the resource is physically deployed) are generated by the FICs, so that the resource can be accessed from the foreign network architectures. As such, the generation of foreign URIs for a given resource was also a target of this evaluation. The performance of such task has a critical impact on the delay of converting messages whenever URIs are required on-the-fly (e.g., while the FIXP is waiting for the URIs to convert the contents towards a foreign network architecture). Although not limited to, in this experiment the following algorithms were used for generating foreign URIs:

- **Concatenation algorithm**, which consisted in replacing the scheme and adding a prefix to the original URI.
- **MD5, SHA-1 or SHA-256 hash algorithms**, which consisted in generating random fixed-size URIs.

The results of this experiment are presented in Figure 3.26, showcasing the delay introduced by each algorithm when generating foreign URIs and how it is impacted by the size of the original URI.

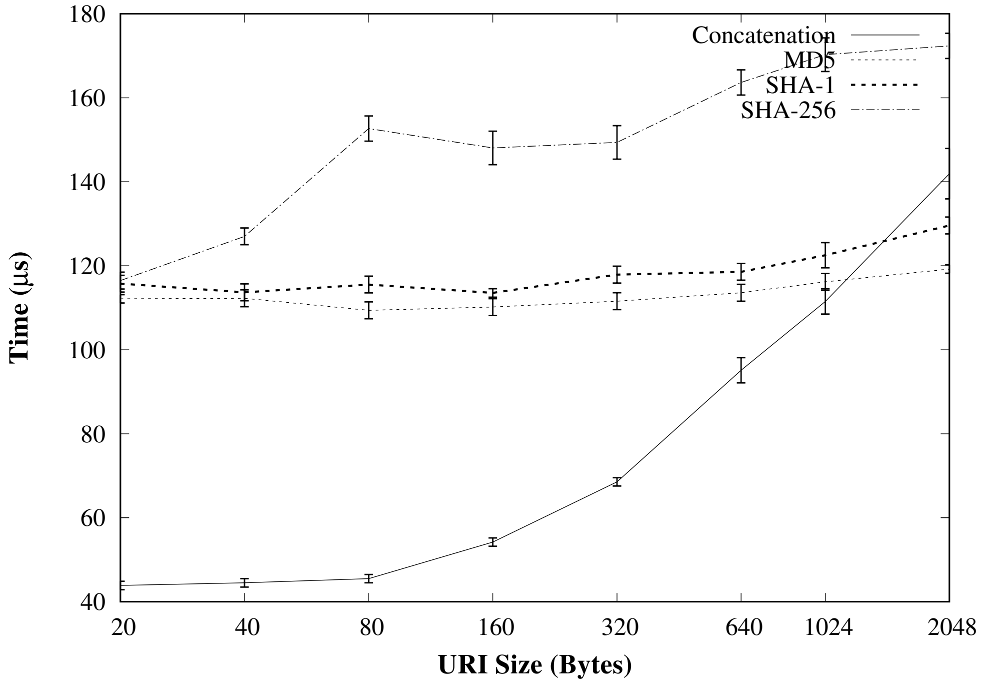


Figure 3.26: Generating foreign URIs for resources

For original URIs up to 1024 bytes in size, the concatenation algorithm introduced a lower delay than any of the algorithms using a hash function. Above this value, the algorithms that used MD5 and SHA-1 achieved a better performance in terms of time than the concatenation algorithm. Comparing the considered algorithms that use hash functions, the MD5 hash algorithm had the best performance in terms of time, while the SHA-256 had the worst performance.

Notwithstanding, the delay introduced by the generation of foreign URIs has a minimal impact on the overall message conversion delay. In this evaluation, it consisted in less than 180μs for original URIs with sizes up to 2048 bytes.

3.6 Discussion

The evaluation presented in Section 3.4 and Section 3.5 provided some insights about the feasibility of the presented solution to enable the interoperation between network architectures following different design choices. Although sharing resources among different architectures may not have optimal performance, it provides a migration path for the introduction of new network architectures, while it eases the development and deployment of applications by abstracting the underlying interoperability aspects. Nevertheless, the proposed solution allows networking mechanisms of a given network architecture (i.e., in-network cache) to be applied to resources deployed in a different architecture, achieving an improvement of the network utilization performance under some conditions.

The following two subsections briefly discuss the major lessons learned during the development and evaluation of the FIFu framework. More specifically, it focuses on the limitations and practical issues that surfaced during the implementation, deployment and evaluation of the proposed framework, which may guide future development stages and further research.

3.6.1 Limitations

A set of limitations on the designed framework which affect its performance were identified. Although some are a direct result from the design choices, others are related with the way each network architecture operates.

Receive complete message before conversion: By acting on behalf of both communication endpoints, the FIXP needs to completely receive messages before being able to convert and to send them towards the destination network architecture (because of e.g. signature generation in the HTTP to NDN and signature verification in NDN to HTTP). However, this implies that additional delay is introduced by the FIXP, besides the delay introduced by the conversion procedure.

(Im)mutable resources and their identifiers: Some architectures define self-certified identifiers for resources, which are identifiers that by themselves allow content integrity to be verified without requiring third-party certification. This implies that identifiers are generated based on known information about the resources (e.g., hash of the content). However, in the presented framework, the FIC infrastructure may generate the identifiers to be used on foreign architectures based only on the original URI of the resource and, therefore, the FIC may not have the required information to generate a correct self-certified identifier. Thus, an immutable resource is treated as being a mutable resource in foreign network architectures. Complementary security mechanisms, supported by the foreign network architecture, may be required in order to achieve a security level (e.g., integrity and authenticity) for a given resource compliant to that offered by its original network.

Even so, identifiers can be updated dynamically as soon as the proper self-certified identifier could be generated. However, this must be done carefully since it could invalidate existing links to the old identifier. The security implications of this strategy are an important aspect, but fall out of the scope from this thesis.

Anchoring-related issues: With traffic destined to a different network architecture requiring to go through the FIXP, anchoring-related issues (such as, longer and/or sub-optimal forwarding paths) are likely to occur. The deployment of FIXPs near to the network edges could mitigate these issues, but on the other hand more complex networks (due to the increasing number of FIXPs) need to be managed.

Sub-optimal support of features: Features supported by a given network architecture (e.g., security, in-network caching and push-communications) may not be natively supported by other architectures. Thus, workarounds to support those features are required, resulting on the sub-optimal support of features provided by different architectures. For example, in network architectures that operate under a request/response communication model, the support of publish/subscribe capabilities may be possible through a periodic polling, whose configuration affects the freshness of the information or signaling overhead in the network.

Security and trust: Although security was not directly addressed in this work, it is an essential aspect in today's Internet and, therefore, further research is required. Notwithstanding, the main points this framework infrastructure needs to take into consideration were identified: (i) the FIFu infrastructure should be trustworthy to all participants (e.g., network operators, resource providers and clients); (ii) the communication between FIXPs and FIC must be secure; (iii) since the FIXPs act on behalf of both endpoints of the communication, secure communications, if required, must be replicated by the FIXPs using relevant certificates and keys. Regarding the latter, solutions similar to those applied to Content Delivery Networks (CDNs) [72] can be applied, where e.g. resource providers delegate trust to the FIFu framework. Moreover, security and privacy aspects may be implemented in different ways on each network architecture and, therefore, during message conversion the same security level must be replicated by the FIXP using mechanisms supported by each network architecture.

Scalability: Scalability concerns regarding the proposed framework can be originated due to (i) the increasing number of resource mappings between different network architectures; and (ii) the increasing number of conversion of messages between different network architectures. To cope with the increasing number of mappings, the proposed framework provides an 3-level hierarchical resolution mechanism (as depicted in Figure 3.6). Entities on each level can be replicated not only to allow load-balancing for incoming requests but also to provide redundancy. While mappings are created in the second level (i.e., Network-architecture level), mapping information is stored in the URI level of the proposed resolution mechanism. Depending on the network architecture, in the URI level, mappings can be stored and resolved following a hierarchical strategy or using DHTs. To cope with the increasing number of message conversion at the FIXP, both vertical and horizontal scaling strategies can be applied, since the state information of each FIXP instance has a local scope (i.e., the FIXP holds no state besides the cached mappings and the ongoing messages conversions). As such, following a vertical scalability strategy a given FIXP entity can be empowered with more computing resources and bandwidth or following a horizontal scalability strategy new FIXP instances can be created to load-balancing the conversion requests.

3.6.2 Reality check: Practical issues

During the implementation, deployment and evaluation a set of practical issues arose that hindered the operation of the presented framework as it was initially designed. These aspects are presented in this section.

Fetching dynamic resources in web pages: In static web pages, resources are identified within the HTML document, enabling the FIXP to identify, directly from the HTML, URIs for other resources (e.g., images and style files) and to map them into URIs compatible with other network architectures. As such, in the web browsing use case, all resources belonging to static web pages deployed in the current IP architecture were able to be fetched. In turn, in dynamic web pages, resources may not be explicitly identified. Instead, resource identification may rely on the execution of one or more pieces of code embedded in the web page (e.g., JavaScript). In these cases, some challenges arose while using the presented framework, since it was not able to identify those resources and, therefore, it was not able to create and provide compatible URIs with the foreign network architectures.

What type of resources are convertible: Studying the cases where content conversion is suitable and how such conversion impacts the actual content is a key aspect for future research. Deciding whether or not to convert resource identifiers inside content can be divided into:

- **Security issues:** For example, converting identifiers inside content invalidates integrity checking mechanisms enforced by the resource provider (e.g., content hash and signatures). To mitigate this issue, the FIXP may act as a trusted entity enforcing its own integrity check mechanisms.
- **Technical issues:** For example, due to the fact that the FIXP may not recognize the content type or its semantics, thus being unable to change the identifiers within the content.
- **Management issues:** For example, consumer and providers of resources may want to retain the original content.

Lack/maturity of real applications: The years of evolution and maturity of the current IP architecture allowed the definition of a multitude of well-defined protocols for different use cases. In the case of new network architectures, specific and/or mature higher-layer protocols targeting different use cases was still lacking. Thus, in this work the protocols in the IP architectures were mapped with the generic operation of the studied ICN architectures. The existence of consolidated mechanisms/protocols for the different use cases for each network architecture may enable an easier and more efficient mapping and retrieval of resources across different architectures.

- **Video parameters in live video streaming:** During the implementation of the live video streaming use case, the question arose on which parameters should be used for providing the video via the RTP stream on the IP network architecture. If, in an interaction between PURSUIT and NDN, each *PURSUIT Publish Data* message was mapped into a new NDN chunk, the lack of a well-defined and mature higher-level protocol (e.g., a protocol to establish or control media sessions) hindered the session setup of the RTSP session, since the video parameters were unknown during this setup process. As such, the FIXP freely defined the parameters to be used (independently of the real parameters of the received video stream) and re-encoded it to match the parameters defined during the RTSP session establishment. However, by doing this, the original video stream is being modified, which is not desirable in a general solution.

- **User-specific content retrieval:** The same resource may vary depending on the user requesting it (e.g., premium access to contents), information that usually goes along with the request as metadata (e.g., cookies in HTTP protocol). The lack/maturity of higher-level protocols in the new network architectures that support this information to go along with the request also imposes a new problem while using the presented framework. The presented framework needs to handle the resources for each user as being different resources, which not only impacts the scalability of the solution, but also affects the control of which users have access to each resource when accessed by a foreign network architecture.

Note that these practical issues can in some way be expected to be solvable in production-level code and more mature Future Internet architectures.

3.7 Concluding Remarks

In this chapter, a way on how to enable the interoperation between different network architectures was presented. Specifically, a novel interoperability framework for the Future Internet, named FIFu (Future Internet Fusion), was presented. FIFu allows the current Internet and new (clean-slate) network architectures to share resources between one another, in a transparent way to end-user, network devices and applications. In doing so, each network architecture can be developed and evolved independently of the others. Such framework leverages the coexistence of different network architectures, which can be seen as an evolution strategy that will further facilitate the initial roll out of new network architectures by promoting its incremental deployment over time.

This framework was implemented and evaluated through simulation, in a web browsing use case, and through a proof-of-concept prototype, in a web browsing, live video streaming and video on-demand scenarios. These scenarios allowed to demonstrate the feasibility of the proposed framework in enabling resource sharing among different network architectures, as well as the impact that the inherent conversion procedures have in the overall communication when accessing resources deployed in a different network architecture. Results showed that it was able to achieve a similar performance when compared with native approaches in the evaluated scenarios. Moreover, the provision of a given resource on the foreign network architectures allowed the capabilities of each architecture (e.g., caching and request aggregation) to be applied to the resource, as if it is originally deployed on each network architecture. However, a set of issues arose during the implementation and evaluation of the proposed framework, such as the anchoring-related issues, sub-optimal mapping of features among different architectures and concerns about content compatibility across different network architectures. Nevertheless, some of the encountered issues are expected to be mitigated as each network architecture evolves and matures with new protocols and mechanisms targeting specific use case scenarios.

Chapter 4

Supporting Mobility in Future Internet Architectures

In the previous chapter a framework that enables entities to access resources deployed in different architectures was proposed, which may leverage the incremental deployment of new network architectures. Nevertheless, mobility in these architectures is still a key requirement that needs to be addressed on each individual architecture for its adoption being considered as a potential Future Internet architecture. Thus, this chapter addresses this subject and contributes to the development of mobility management mechanisms in both evolutionary SDN-based and clean-slate ICN network architectures, as well as in scenarios where mobility crosses different network architectures.

4.1 Introduction

The increasing proliferation of mobile devices equipped with multiple wireless interfaces (such as, smartphones) created heterogeneous scenarios where network operators need to provide connectivity for different kinds of access networks. At the same time, it opens space for a new set of opportunities taking advantage of the different technologies to provide an always best connected experience to the user. In this way, the capability of handling mobility efficiently in Future Internet network architectures is a key requirement that needs to be addressed by those architectures.

Furthermore, the traffic generated by mobile devices is continuously increasing and, by 2021, traffic from wireless and mobile devices will account for more than 63% of the total IP traffic [25]. The dynamics of the wireless environments, and the associated heterogeneity, pose a complex challenge for assuring that the content reaches the user in an optimized way.

As such, supportive mechanisms to ensure that the content is sent to the requester in the most optimal way (i.e., best available point of attachment or best suitable wired/wireless technology) are required. Aspects related to the conditions of the wireless networks, both the currently connected or neighboring networks, are factors that can help to decide how to best deliver the content to the user. Considering this scenario, IEEE developed the IEEE 802.21 standard for Media Independent Handover [1]. Its main purpose is to facilitate and optimize inter-technology handover processes by providing a set of media-independent primitives, thus creating an abstraction regarding the link layer, not only for obtaining link information and controlling link behavior, but also for assisting the different steps of the handover procedure.

In this chapter, the mechanisms defined in IEEE 802.21 are integrated into two different frameworks, one focusing on SDN-based and the other on ICN architectures. By being aware of the link conditions at each moment and by having control over the different phases of the handover procedure, the proposed frameworks optimize the handover procedures in both Future Internet architectures, decreasing handover delay and packet loss. Additionally, with the edge of the network being a potential location for the initial deployment of new network architectures, an heterogeneous environment where access networks support either IP and/or ICN architectures can be expected. In doing so, a novel mobility framework that tackles the mobility of MN across different network architectures is proposed, allowing resources to continue being reached even if deployed in a different network architectures after the handover of MNs.

4.2 Supporting Mobility in Evolutionary SDN-based Architecture

As part of the work developed in this thesis, the controlling mechanisms of SDN architectures were enhanced with IEEE 802.21 mechanisms in a single framework, with the purpose of enhancing the operation of evolutionary SDN-based architecture with the capability of efficiently managing mobility procedures. The main driver that motivated the integration of both SDN and IEEE 802.21 mechanisms was to enable an SDN infrastructure to deal with different wireless access networks in an abstract way, becoming capable of taking advantage of each one of these networks to optimize the overall performance of the network. While SDN abstracts the network control by providing a common API, IEEE 802.21 brings a similar concept by abstracting link layer technologies. With this integration, controlling entities are able (i) to (re)program the network on-the-fly by using SDN mechanisms; and, through the use of IEEE 802.21 mechanisms, (ii) to acquire context information from the link-layer of wired and wireless interfaces of both network entities (such as, PoAs and MNs); and (iii) to trigger and facilitate handovers control.

By combining both mechanisms, controlling entities not only have a better knowledge of the network state, allowing them to make better decisions to optimize the overall performance and resource usage of the network as well as to improve MNs' connectivity by triggering handovers to more suitable networks.

In doing so, the main goal of the proposed framework is to have a first assessment of the benefits that SDN could have if aware of the both wired and wireless conditions of the network under its control.

4.2.1 Framework Overview

An overview of the proposed IEEE 802.21-enhanced SDN framework is presented in Figure 4.1, where SDN is instantiated in the form of the OpenFlow protocol, along with OpenFlow-enabled switches and controller. Instead of only being responsible for controlling nodes in the core network, this framework envisages OpenFlow being used to control network flows up to the network side endpoint, affecting, as such, wireless network PoAs (e.g., WLAN Access Point). Still, OpenFlow mechanisms are not aware of specificities of the wireless link, being only capable of (re)configuring the flow tables of such devices. To fill this gap, the proposed framework combines OpenFlow with IEEE 802.21, empowering it with mechanisms to acquire context information about the link layer of different technologies (e.g., receive events regarding current link status perceived by the MN and PoAs) as well as to provide mechanisms to assist and facilitate handover control procedures (e.g., check resource availability, and prepare, commit and release resources). Management and controlling entities (e.g., mobility management entities) can then use these mechanisms to acquire information about the wireless environment and, based on that information, (re)configure the network to achieve better performance or even trigger the handover of specific nodes to e.g. load-balance existing MN among several PoAs or handover the MN to a better PoA. In addition, the media-independent behavior of IEEE 802.21 enables the framework herein proposed to be technology agnostic, allowing the mechanisms defined here to be deployed over different link technologies.

Summarizing, the controlling entity, acting as the OpenFlow controller, communicates via OpenFlow protocol with the underlying network entities from the core up to the edge of the network and via MIH protocol of the IEEE 802.21 standard, acting as an IEEE 802.21 Point of Service, not only with core and edge network entities, but also with MNs connected to the PoAs under its domain.

Through the combination of both technologies, the proposed framework enables network management scenarios, where flows of a given MN can be dynamically and

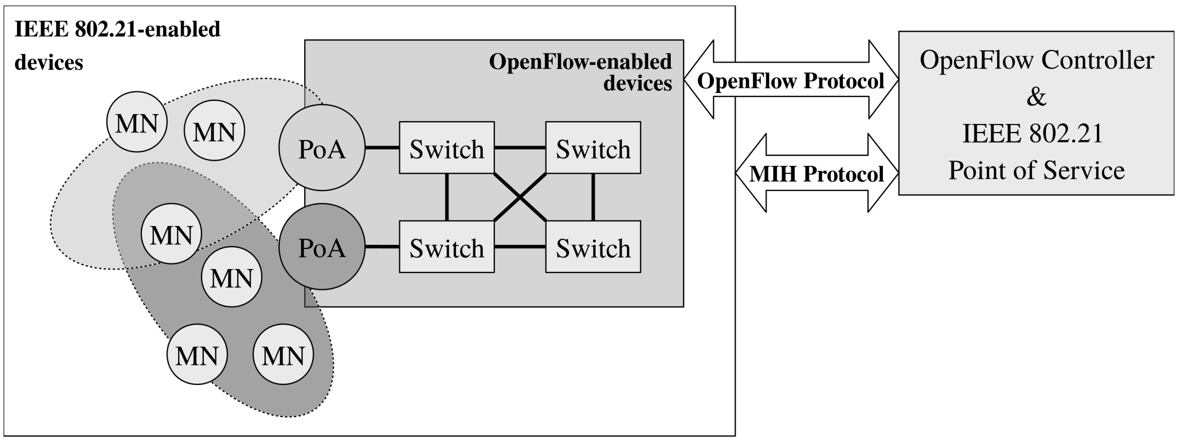


Figure 4.1: Proposed IEEE 802.21-enabled SDN framework

preemptively configured according to the wireless connectivity opportunities detected by the MN while it moves. In doing so, paths can be establish before the MN moves towards the new location (i.e., flows being sent towards the new PoA before the MN handover), reducing the impact on the data sessions involving the MN.

Finally, the flexibility of this framework allows the integration with different mobility protocols (e.g., PMIPv6), allowing it to accommodate different IP mobility procedures when needed.

4.2.2 Network Entities

The proposed framework (Figure 4.1) features three different entities: (i) an OpenFlow controller enhanced with IEEE 802.21 Point of Service capabilities; (ii) OpenFlow Switches (representing both core and PoA); and (iii) mobile nodes. Next, a detailed operation of each one of these entities is presented, focusing on the enhancements required in what concerns mobility aspects.

4.2.2.1 OpenFlow Controller & IEEE 802.21 Point of Service

This is the central entity of the proposed framework (whose internal architecture is depicted in Figure 4.2) responsible for controlling the overall operation of the underlying network.

Regarding mobility aspects, it contains a *Mobility Management* module that handles the mobility procedures of MNs connected to the network under its domain. For that, it acts as an IEEE 802.21 PoS for assisting and controlling the mobility procedures. This includes tasks such as checking the available resources, used as input for deciding the best handover candidate PoA(s), which is then notified to the MNs. In addition, it also includes tasks related with the preparation and reservation of resources

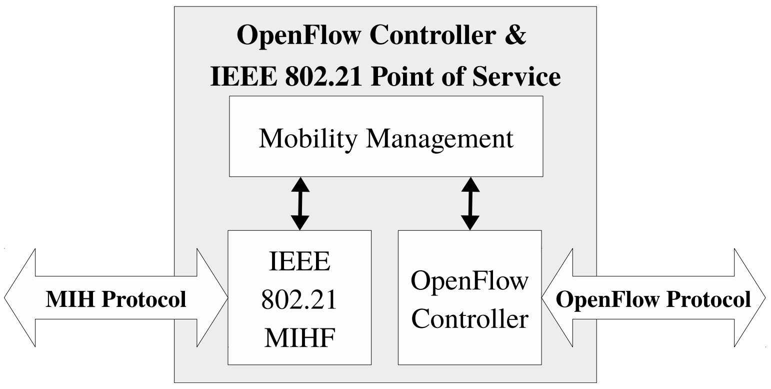


Figure 4.2: OpenFlow Controller & IEEE 802.21 Point of Service internal architecture

before the handover occurs and resources release after the handover procedure. The communication with the MNs and PoAs is performed through the MIHF using the MIH protocol.

The *Mobility Management* module is also responsible for triggering the mechanisms to adapt the network before and after the actual handover procedure. For that, the *Mobility Management* module requests the OpenFlow Controller to perform a set of routing related tasks (i.e., updating the forwarding tables of OpenFlow Switches). More specifically, using the OpenFlow protocol, it reprograms the network to (i) send the traffic related to the MN not only to its actual PoA but the expected future PoA before the handover procedure; and (ii) stop sending the traffic related to the MN towards the old PoA after the handover procedure. This procedure is further described in Section 4.2.3.

4.2.2.2 OpenFlow Switch (& Point of Attachment)

This entity can represent either a switch in the core network or a point of attachement. The behavior and internal architecture (as depicted in Figure 4.3) herein described is the same for both cases, except that the latter is deployed on the edge of the network providing link connectivity to end-user equipment (e.g., mobile nodes) to the network.

Besides its operation as a standard OpenFlow switch, this entity is enhanced with IEEE 802.21 capabilities. Regarding the former, it executes data packet forwarding based on the rules stored in flow tables configured by the OpenFlow Controller using OpenFlow protocol. For supporting the latter capabilities, the standard OpenFlow switch architecture is enhanced with three new modules: (i) a *Link SAP*, for each network interface, responsible for collecting context information about the link-layer of the corresponding interface as well as to control and apply actions to that same interface; (ii) *Mobility Management* module that is responsible for assisting the handover procedures and to monitor its network interfaces by requesting context

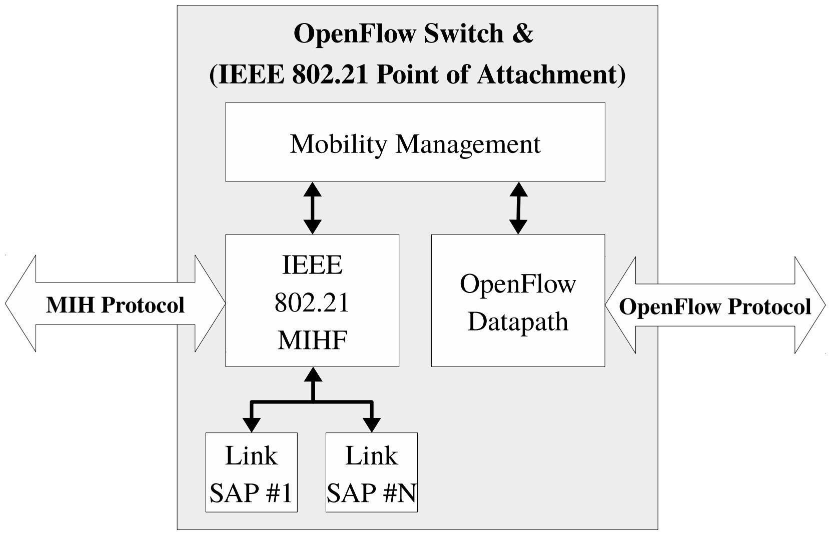


Figure 4.3: OpenFlow Switch (& Point of Attachment) internal architecture

information from the *Link SAPs*; and (iii) an MIHF that behaves as a middleware allowing the communication via MIH protocol between local or remote *Mobility Managements* and the *Link SAPs* or between remote *Mobility Managements*.

Leveraging the IEEE 802.21 capabilities, the PoA is able to trigger events about the link conditions of its interfaces, sending them towards its (or a remote) *Mobility Management* entity. For example, it can trigger an event towards the *Mobility Management* in the PoS whenever a new MN is connected, which, detecting that the a given PoA is being overloaded with too many MNs, initiates the handover of some of those MNs to a nearby and less overloaded PoA.

4.2.2.3 Mobile Node

The MN represents the end-user equipment (e.g., laptops and smartphones) through which users connect to the network. This equipment can be equipped with one or more access technologies, either wired (e.g., Ethernet) or wireless (e.g., WLAN and 3G). As depicted in Figure 4.4, the MN is an IEEE 802.21-enabled device being, therefore, equipped with a MIHF to allow the remote communication with the PoS via MIH protocol. Besides the MIHF, the MN is equipped with *Link SAPs*, one for each network interface, which are responsible for tasks such acquiring context information about the link layer of the corresponding network interface. Finally, a *Mobility Management* module is also deployed in the MN responsible for monitoring the link conditions of the underlying network interfaces, triggering and assisting the handover procedures whenever required.

Using the IEEE 802.21 mechanisms, the MN can report events whenever it e.g. detects a new PoA or when the signal level of its current connection crosses a predefined

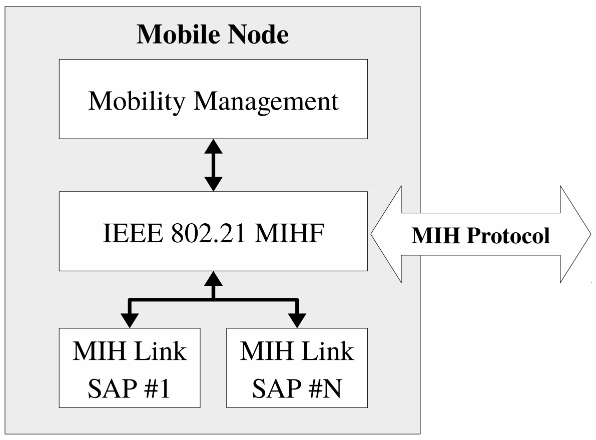


Figure 4.4: Mobile Node internal architecture

threshold. Upon the reception of such events, the *Mobility Management* in the PoS or in the MN can initiate the handover procedure towards another PoA.

4.2.3 Handover Scenario Signaling

The link layer abstraction enabled by IEEE 802.21 allows this framework to be decoupled from link related aspects. As such, the presented signaling in this section can be deployed over different technologies (e.g. WiFi, WiMAX and mobile).

The proposed framework approaches the handover procedure in three steps: (i) handover preparation; (ii) handover commit; and (iii) handover complete. As the name indicates, the first step is related with the preparation procedures before the occurrence of the handover itself, which includes checking the available resources in the possible handover candidates. In the second step, the MN commits to handover to the chosen PoA, with resources in the handover target (and eventually, other entities of the network) being prepared to accommodate the MN. In addition, the handover procedure itself is executed at the end of this step. Finally, the last step is related with conclusion aspects, such as the release of resources from the old PoA.

The initiation of each step of the handover procedure could be triggered by the MN or the network (by both the Controller/PoS or the PoA). The signaling presented in the following subsections encompasses all possible scenarios. In the dashed boxes, the signaling specific to each case is represented, where A, B and C identify if the trigger is initiated by the MN, Controller/PoS and PoA respectively. The remaining signaling is common to all the cases.

4.2.3.1 Handover Preparation

The proposed framework is flexible to allow different entities to trigger the initiation of the handover procedures. The controlling entity, which by monitoring the overall state of the network at each instant, can decide to handover specific flows or MNs to another PoA due to e.g. load-balancing purposes. Additionally, the MNs can request the initiation of the handover procedure if e.g. better connectivity conditions are found. The PoA can also trigger the initiation of the handover procedure, although indirectly, by sending events to the PoS indicating that it is e.g. overloaded. The signaling for each case is illustrated in Figure 4.5.

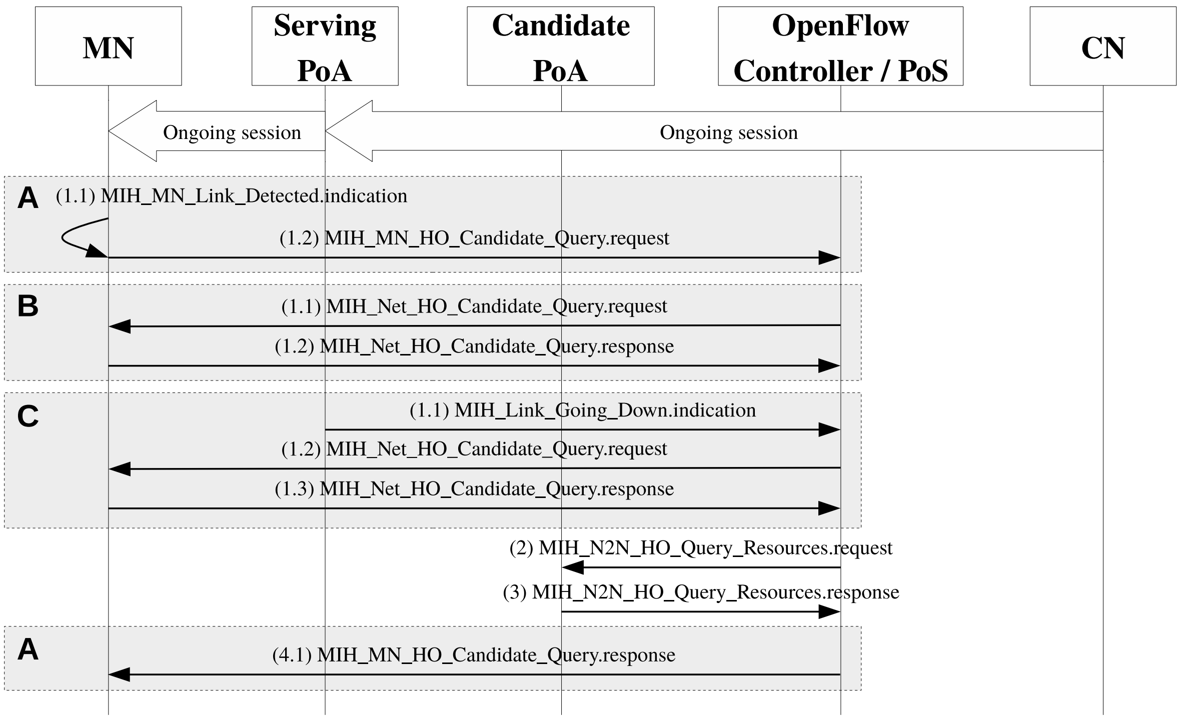

MN
Serving PoA
Candidate PoA
OpenFlow Controller / PoS
CN
Ongoing session
Ongoing session
A
(1.1) MIH_MN_Link_Detected.indication
(1.2) MIH_MN_HO_Candidate_Query.request
B
(1.1) MIH_Net_HO_Candidate_Query.request
(1.2) MIH_Net_HO_Candidate_Query.response
C
(1.1) MIH_Link_Going_Down.indication
(1.2) MIH_Net_HO_Candidate_Query.request
(1.3) MIH_Net_HO_Candidate_Query.response
(2) MIH_N2N_HO_Query_Resources.request
(3) MIH_N2N_HO_Query_Resources.response
A
(4.1) MIH_MN_HO_Candidate_Query.response

Figure 4.5: Handover Preparation Signaling using the proposed IEEE 802.21-enabled OpenFlow Framework

While on the move, the MN can detect new PoAs in its surroundings, triggering an event to its mobility manager notifying about the detected PoAs. The mobility manager in the MN can then decide based on several factors (e.g., signal strength of the PoAs under its range) sending a *MIH_MN_HO_Candidate_Query.request* (message A1.1 in Figure 4.5) towards its PoS, indicating the potential PoAs that it is interested in moving to. By doing so, the MN is triggering a mobile-initiated handover.

In turn, the mobility entity in the PoS can trigger a network-initiated handover by sending a *MIH_Net_HO_Candidate_Query.request* towards the MN with a list suggesting potential PoAs for the MN handover to (message B1.1 in Figure 4.5). The MN responds with a *MIH_Net_HO_Candidate_Query.response* (message B1.2 in Figure 4.5), acknowledging not only the handover initiation, but also informing the PoS about the preferred PoAs.

The third possibility is the initiation of the handover by the PoA. For example, the PoA can trigger an event to the PoS indicating that one of its interfaces is going down (e.g., due to malfunction) (message C1.1 in Figure 4.5). Upon the reception of such event, the PoS triggers the handover of all the MNs connected to that PoA using the same signaling as describe before for the network-initiated handover (message C1.2 and C1.3 in Figure 4.5). In this case, the PoS can handle the handover of each MN individually, or handle it as a single handover of a group of MNs [27, 6, 5].

Independently of the chosen approach, at this stage, the PoS holds information regarding the potential candidate handover PoAs. As such, for each of these PoAs, the PoS checks available resources (messages 2 and 3 in Figure 4.5) to verify if they can accommodate the attachment of the MN.

Finally, for the mobile-initiated handover use case, the PoS returns a list of potential candidate PoAs to the MN, sorted from the most preferred to the least preferred by sending a *MIH_MN_HO_Candidate_Query.response* (message A4.1 in Figure 4.5).

4.2.3.2 Handover Commit

The handover commit step can be initiated by either the MN or the PoS (as depicted in Figure 4.6).

Whenever this process is initiated by the MN, it starts by notifying the PoS about the selected PoA by sending a *MIH_MN_HO_Commit.request* to the PoS (message A1.1 in Figure 4.6). If initiated by the PoS, the MN's decision corresponds to the most preferred PoA indicated by the MN in the handover preparation step (message A4.1 in Figure 4.5).

Upon the reception of the MN's decision, the PoS reserves the resources at the target PoA to accommodate the incoming MN (messages 2 in Figure 4.6). When the PoA acknowledges the PoS request with a sucessful status (message 3 in Figure 4.6), the PoS proceeds to the reconfiguration of the network in order to make ongoing sessions of the MN reach the candidate PoA. For that, using OpenFlow protocol (i.e., acting as the OpenFlow Controller), it issues a *OFPT_FLOW_MOD* (message 4 in Figure 4.6) message towards the PoA as well as to other OpenFlow switches under its domain that require an update of the forwarding tables. An *OFPT_BARRIER_REQUEST* message (message 5 in Figure 4.6) is sent after the *OFPT_FLOW_MOD* message, so that when the corresponding response (message 6 in Figure 4.6) is received, the PoS knows that the forwarding rules were already configured. At this point, the path to the candidate PoA is established before the handover procedure occurs, aiming to reduce the time

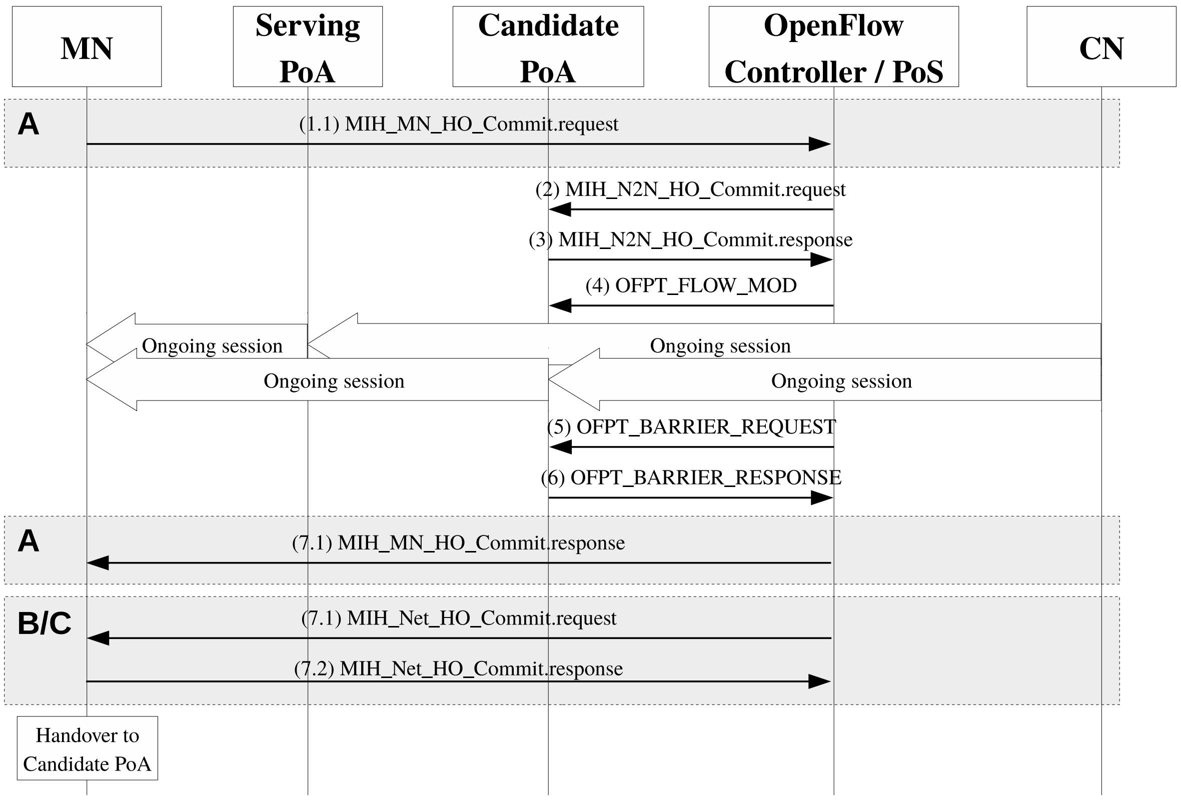

MN
Serving PoA
Candidate PoA
OpenFlow Controller / PoS
CN
A
(1.1) MIH_MN_HO_Commit.request
(2) MIH_N2N_HO_Commit.request
(3) MIH_N2N_HO_Commit.response
(4) OFPT_FLOW_MOD
Ongoing session
Ongoing session
Ongoing session
Ongoing session
(5) OFPT_BARRIER_REQUEST
(6) OFPT_BARRIER_RESPONSE
A
(7.1) MIH_MN_HO_Commit.response
B/C
(7.1) MIH_Net_HO_Commit.request
(7.2) MIH_Net_HO_Commit.response
Handover to Candidate PoA

Figure 4.6: Handover Commit Signaling using the proposed IEEE 802.21-enabled OpenFlow Framework

required to restore ongoing sessions after the handover itself. Nevertheless, the path to the old PoA is temporarily kept in order to maintain the ongoing sessions before the handover occurs.

At this stage, the selected candidate PoA is ready to accommodate the MN and, therefore, the PoS instructs the MN to initiate the L2 handover procedure to the candidate PoA. If the handover commit step was initiated by the MN, the PoS sends an *MIH_MN_HO_Commit.response* message to the MN (message A7.1 in Figure 4.6). Otherwise, the PoS notifies the MN about the target PoA via *MIH_Net_HO_Commit.request* message (message B/C7.1 in Figure 4.6), which, in turn, acknowledges its reception by sending an *MIH_Net_HO_Commit.response* message (message B/C7.2 in Figure 4.6) to the PoS.

In the end of handover commit step, the MN executes the L2 handover to the selected candidate PoA.

4.2.3.3 Handover Complete

The following step is the handover completion, in which resources allocated to the MN in its old locations are released. The signaling presented in Figure 4.7 is the same independently of which entity initiated the handover procedure.

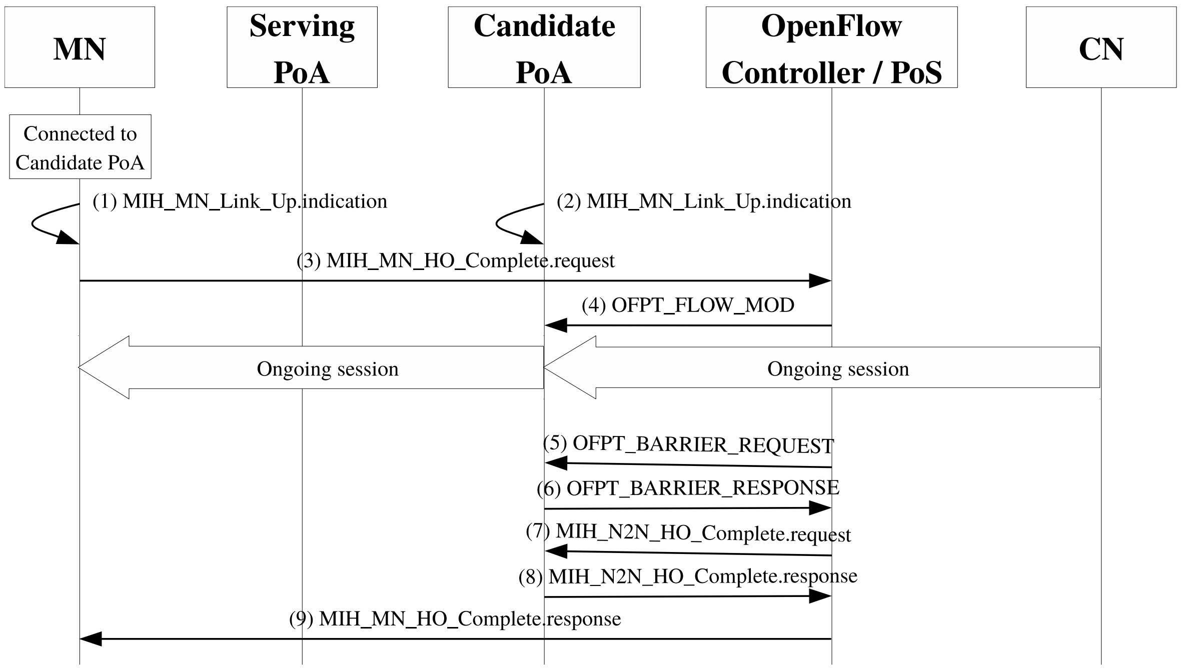

MN
Serving PoA
Candidate PoA
OpenFlow Controller / PoS
CN
Connected to Candidate PoA
(1) MIH_MN_Link_Up.indication
(2) MIH_MN_Link_Up.indication
(3) MIH_MN_HO_Complete.request
(4) OFPT_FLOW_MOD
Ongoing session
Ongoing session
(5) OFPT_BARRIER_REQUEST
(6) OFPT_BARRIER_RESPONSE
(7) MIH_N2N_HO_Complete.request
(8) MIH_N2N_HO_Complete.response
(9) MIH_MN_HO_Complete.response

Figure 4.7: Handover Complete Signaling using the proposed IEEE 802.21-enabled OpenFlow Framework

Upon the connection to the candidate PoA, an event is generated either on the MN (message 1 in Figure 4.7) and on the candidate PoA (message 2 in Figure 4.7). The MN, detecting that the connection to the candidate PoA is already established, informs the PoS by sending an *MIH_MN_HO_Complete.request* (message 3 in Figure 4.7) which, in turn, initiates the procedures to release the resources allocated to the MN in what concerns its old location. Thus, the PoS triggers the OpenFlow procedures to clear routing information related with the MN from the old PoA, by updating the flow tables of both the old PoA and intermediate switches (message 4 in Figure 4.7). As in the previous step, *OFPT_BARRIER_REQUEST* messages (message 5 in Figure 4.7) are sent after the *OFPT_FLOW_MOD* so that whenever it receives the corresponding *OFPT_BARRIER_REPLY* messages (message 6 in Figure 4.7) it knows that the flow tables have been updated. In parallel, the PoS also notifies the old PoA to release the resources allocated to the MN (messages 7 and 8 in Figure 4.7). Finally, the PoS acknowledges the MN that the handover procedure on the network side is completed by sending a *MIH_MN_HO_Complete.response* message (message 9 in Figure 4.7).

4.2.4 Evaluation

In order to evaluate the feasibility of the proposed IEEE 802.21-enabled OpenFlow framework, the framework entities as described in Section 4.2 were implemented using

ODTONE[1] [28] (an open-source implementations of both IEEE 802.21), OpenFlow 1.3 Software Switch[2] (an open-source implementation of the OpenFlow Switch) and NOX OpenFlow 1.3 Controller[3] (an open-source implementation of the OpenFlow Controller).

4.2.4.1 Testbed Description

For the evaluation of the proposed framework, the scenario depicted in Figure 4.8 was deployed over the AMazING [16] wireless network testbed, located on the rooftop of Instituto de Telecomunicações - Aveiro building.

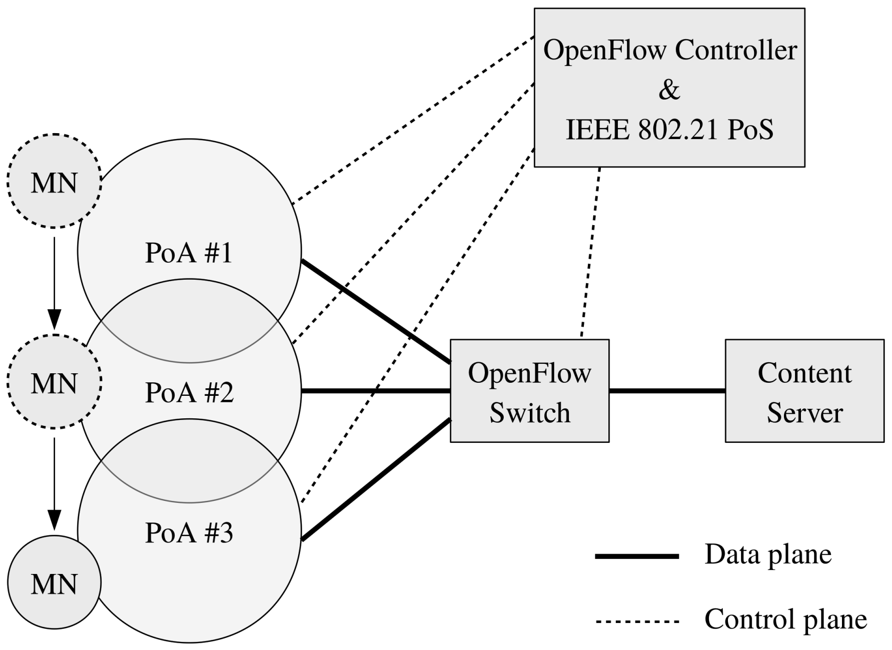


Figure 4.8: Scenario description for evaluation of the IEEE 802.21-enabled OpenFlow framework

The evaluation scenario is composed by three different PoAs, each one serving a different IPv6 network, connected to a common switch. Both the PoAs and the switch are OpenFlow and 802.21-enabled, being monitored and controlled by the *OpenFlow Controller & PoS* entity via OpenFlow and MIH protocols. The scenario also encompasses a content server and a MN which, respectively, streams and consumes multimedia content.

Each physical node is configured with a VIA Eden 1GHz processor with 1GB RAM, a 802.11a/b/g/n Atheros 9K wireless interface, and a Gigabit wired interface. Each node runs the Linux OS (Debian distribution) with kernel version 3.2.0-2-686-pae.

[1]Open Dot Twenty ONE (ODTONE) - https://atnog.av.it.pt/odtone
[2]OpenFlow 1.3 Software Switch - https://github.com/CPqD/ofsoftswitch13
[3]NOX OpenFlow 1.3 Controller - https://github.com/CPqD/nox13oflib

In the evaluation scenario, the MN is initially attached to PoA #1 receiving a video stream from the content server. At times 10s and 20s it handovers to the PoA #2 and PoA #3 respectively, following a MN-initiated handover approach as described in Section 4.2.3. Since the same interface was used during the handovers, the handovers are break-before-make. The MN is expected to continue to receive the video stream from the content server while on the move.

Each experiment was run 10 times and averaged results with a 95% confidence interval are shown. In order to correlate the results between runs, handovers start at the same time between experiments.

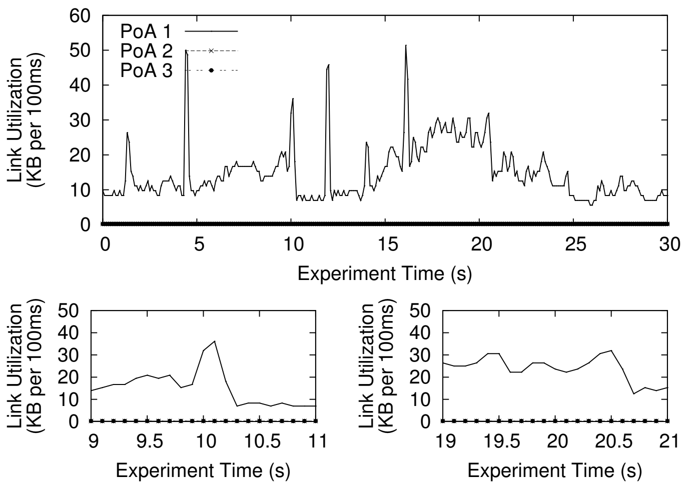
4.2.4.2 Performance Evaluation

In this section, the performance of the proposed framework was evaluated, being compared with (i) a deployment without any mobility-aware mechanism; and with (ii) a deployment featuring some enhanced logic on the OpenFlow controller, where a dummy mobility trigger after the handover initiates OpenFlow procedures (i.e., without any support from IEEE 802.21 mechanisms). Obtained results are presented in Table 4.1 and Figure 4.9.

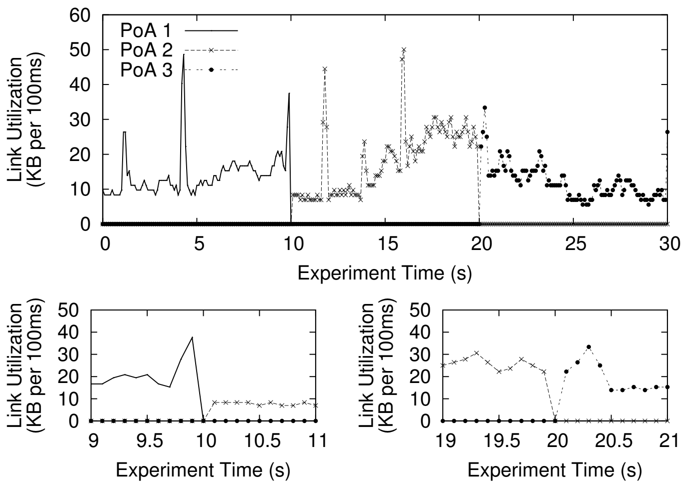
Table 4.1 shows the impact on the video flow reception, including the number of packets sent by the Content Server and received by the MN, percentage of packet lost during the experiment, the delay to restore the video stream after the handover procedure and, lastly, the time during which the video flow was being sent towards a non-required PoA. Figure 4.9 depicts the link utilization during the whole experiment, highlighting the time at which the handover occurs (at time 10s and 20s). In order to clearly observe the behavior during handovers, the results of Figure 4.9 correspond to a single randomly chosen run.

Table 4.1: Packet statistics

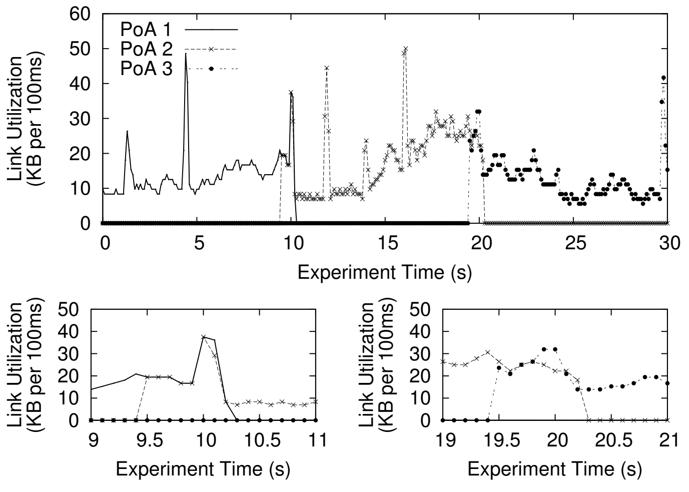
	No mobility awareness	**Dummy mobility**	**IEEE 802.21 supported mobility**
Total sent packets	3346.8 ± 2.1	3218 ± 10.3	3289.7 ± 24.9
Total received packets	950 ± 15.2	3078.4 ± 21.1	3218 ± 30.5
Total packet loss (%)	28.39 ± 0.44	4.34 ± 0.35	2.18 ± 0.33
Lost packets during HO	∞	70.9 ± 11.6	36.95 ± 12.2
Restore stream delay (ms)	∞	433.9 ± 68.2	197.2 ± 62.2
Redundancy (ms)	0 ± 0	0 ± 0	359.0 ± 62.7



(a) No mobility awareness



(b) Dummy mobility



(c) IEEE 802.21 supported mobility

Figure 4.9: Link utilization

Analyzing the results of Table 4.1, it can be observed that the deployment scenario without any mobility-aware mechanism was not able to recover the video stream after the handover. This happened because neither the MN nor the network were able to detect or report the occurrence of handover and, therefore, the path to the new location of the MN was not configured. In addition, the path via PoA #1 (i.e., the initial point of attachment of the MN) was never removed, resulting in wasted usage of the link resources, as seen in Figure 4.9a.

With the dummy mobility trigger, the MN was able to restore the video session approximately 433.9 $\pm$ 68.2 ms after disconnecting from the old PoA. This delay is related not only with the L2 handover procedure, but also with the OpenFlow route update procedures which only happen after the handover and its notification to the PoS. From the total of sent packets, the MN lost about 4.34% packets, mostly during the handover procedure. Packet loss is intrinsically related with the delay in restoring the video stream which, as said previously, corresponds to the time required to establish the connectivity with the new PoA and to setup the route towards the new position of the MN. In terms of packets, it represented an average of 70.9 $\pm$ 11.6 packet lost per handover. Looking at the link utilization graphs (Figure 4.9b), it can be observed that there is no overlap of link utilization by two different PoAs. This behavior occurs because the PoS only establishes the path towards the MN via the new PoA, after releasing the path via the old PoA.

Lastly, from the results of Table 4.1 showing the performance of the presented framework, it can be observed that the handover performance was significantly improved, where packet loss and stream restoration delay were halved when compared to the previous scenario. The MN was able to restore the video session after approximately 197.2 $\pm$ 62.2 ms after disconnecting from the old PoA. Since the route to the new PoA was already setup up before the handover, after the L2 handover the MN started to received the video stream immediately. Therefore, the time required to restore the video stream is only related with the L2 handover procedure delay. Consequently, it decreased the values of packet loss to about 2.18%, which corresponds to approximately 37 lost packets per handover. However, in contrast to the previous deployment scenarios, the video stream is sent towards the old and the new PoA for about 359.0 $\pm$ 62.7 ms per handover. This time is related with the pre-configuration of the path towards the MN via the new PoA (before the handover) and the removal of the path towards the MN via the old PoA (after the handover). Figure 4.9c shows the link utilization overlap before and after the handover procedure. Still, this reflects the conservative nature of the chosen handover management algorithm. The proposed framework is flexible enough to allow the definition of different mobility management strategies, optimizing different KPIs, where link utilization overlap can be one of them.

4.2.4.3 Control Signaling Overhead Analysis

In this section the footprint of the proposed signaling is studied. The obtained results are presented in Table 4.2, showing the results related with the amount of data exchanged and total time required for different phases of the handover process (i.e., preparation, commit and complete). The values for the amount of data exchanged consider not only the size of the MIH or OpenFlow protocols, but also the size of the L4 headers (UDP or TCP).

Table 4.2: Total signaling overhead per handover

		Handover Preparation	**Handover Commit**	**Handover Complete**
Size (bytes)	**IEEE 802.21 (UDP + Ack)**	301	294	329
	OpenFlow (TCP)	0	736	688
Time (ms)	**IEEE 802.21 (UDP + Ack)**	11.73 ± 1.63	128.73 ± 12.41	122.90 ± 0.91
	OpenFlow (TCP)	0 ± 0	56.71 ± 0.85	56.17 ± 0.49

Results from Table 4.2 show that almost 60% of the exchanged signaling corresponds to the OpenFlow protocol and the remaining 40% related with IEEE 802.21. The signaling involving the MN accounts for about 20% of the total signaling overhead, due to its participation in the handover process to assist in the candidate query and handover commit and complete steps.

The overhead of the OpenFlow protocol signaling encompasses not only the *OFPT_FLOW_MOD* messages, which update the forwarding tables in the OpenFlow switches, but also the *OFPT_BARRIER_REQUEST* and *OFPT_BARRIER_REPLY* messages, which are used to assure that the routing rules were configured in the OpenFlow switch. If the status of the flow table update was not verifyed, it would reduce in about 144 bytes the signaling related to OpenFlow. In addition, OpenFlow is sent over TCP, introducing the overhead of the TCP header and the corresponding TCP acknowledgment messages. Also, the OpenFlow protocol overhead is highly dependent on the number of switches on which the PoS should update the forwarding tables. The configuration of each OpenFlow switch would increase the OpenFlow signaling between 344 and 392 bytes for the proposed scenario, depending on the flow filter and actions to perform. Lastly, the signaling related with the OpenFlow protocol does not involve the MN, being exchanged exclusively between the PoS and the new and old PoAs and the intermediary switch. Regarding time, the OpenFlow procedures took about 112 ms per handover to perform the routing update on the OpenFlow switches, divided into 56.71 ± 0.85 ms to preemptively configure the path through the new PoA and 56.17 ± 0.49 ms to release the path through the old PoA.

Regarding the IEEE 802.21 signaling, all signaling involves the PoS, on which about 51% of the exchanged data is related with the communication with the MN, being the remaining 49% exchanged with the PoAs. The old PoA and the new PoA are involved in about 30% and 19%, respectively, of the total IEEE 802.21 signaling. The higher amount of information exchanged with the new PoA highlights its involvement in the resources querying and committing processes, in opposite to the old PoA which is only involved in the releasing of resources. The signaling involving the MN is related with its assistance in the candidate query and handover commit and complete procedures. In terms of time, the handover preparation procedures took about 11.73 ± 1.63 ms, which is minimal compared with the remaining procedures. The handover Commit and handover Complete procedures took about 128.73 ± 12.41 and 122.90 ± 0.91 ms respectively. However, besides the time required to prepare and to release the link resources, these values are highly affected by the OpenFlow procedures, since they are encompassed between the IEEE 802.21 procedures.

4.2.5 Discussion

The dynamic and heterogeneous nature of wireless technologies creates a complex environment to ensure an always best connected experience for MNs, where context information from the link layer (i.e., information about the network interfaces from the MN and surrounding wireless conditions) plays an important role to help achieving such experience for MNs.

IEEE 802.21 provides a set of mechanisms for not only acquiring context information from the link-layers but also managing the different phases of the handover procedure of MNs. As such, with the integration of the IEEE 802.21 mechanisms with the SDN procedures, the SDN controller is able not only to have an enhanced overview of network with additional information about the link-layer of the underlying entities, but also to have the capability of managing the handover procedures of MNs and, consequently, reconfigure the underlying network according to the movement of MNs. In doing so, the SDN controller can reconfigure the network to cope with the new location of the MN or according to predictions on the MN movement before the MN actually moves to its new PoA.

In the scenario used to exemplify and evaluate the proposed mobility framework, the SDN Controller preemptively reconfigured the network to forward the MN's flows towards its new PoA before the handover and to release the forwarding of MN's flows towards the old PoA only after the handover is complete. Such strategy aimed to mitigate the service interruption of the MN due to the handover procedure, but it had the drawback of sending the MN's flows towards two different locations simultaneously during the handover procedure. Nevertheless, this is just one of the potential use cases of the proposed framework, which is flexible enough to accommodate different strategies aiming to improve different KPIs. For example, in a scenario where multiple MNs are accessing the same multicast stream, the SDN Controller is able to instruct nodes to move to the same PoA (provided that MNs are on the range of the same PoA) in order to exploit the advantages provided by multicast and make optimal use of the available bandwidth in the network.

4.3 Supporting Mobility in a Clean-slate ICN Architecture

With mobility having a great impact on nowadays communications, mobility becomes a key requirement that should also be addressed by clean-slate ICN architectures, before being acknowledged as potential candidates for the Future Internet. In this thesis, a specific ICN architecture is selected (i.e., ETArch), but the concepts defined below can be adapted to cope with other ICN instantiations.

In doing so, this section focuses on Entity Title Architecture [33], a clean-slate SDN-based ICN architecture, enhancing its operation to cope with the capability of handling mobility of ETArch-enabled MNs. As such, likewise the solution described in the previous section, ETArch mechanisms were integrated with IEEE 802.21 mechanisms in a single framework, allowing the controlling entity of the ETArch architecture (i.e., the DTSA) to take advantage of the different wireless access networks to optimize the overall performance of the network in a technology-abstracted way. The DTSA becomes capable of (i) controlling and managing the handover procedures of the MNs supporting the ETArch architecture; and (ii) acquire context information from the link layer of wired and wireless interfaces of both network entities (such as, PoAs) and MNs; while retaining its capabilities of controlling and managing the underlying network by means of SDN mechanisms.

By combining both mechanisms, controlling entities have a better knowledge of the state of the underlying network, allowing them to optimize the workspace configuration and to improve MNs' connectivity by assisting their handover to more suitable networks. For example, besides triggering the handover of MNs due to link connectivity aspects (e.g., signal strength of the current connection of the MN gets weaker), the DTSA may trigger the handover of a given MN to a PoA that is already providing towards its access network the same workspaces subscribed by the MN, with the purpose of leveraging the multicast capabilities of the ETArch architecture and, consequently, optimizing the usage of the network resources.

In doing so, the main goal of the proposed framework is to provide the DTSA with supportive mechanisms to become aware of the link specificities of both network entities and MNs as well as of being capable of handling the mobility of MNs, providing a first assessment of the impact that it can have in the overall performance of the network.

4.3.1 Framework overview

The framework proposed in this section aims to enhance the ETArch architecture with IEEE 802.21 mechanisms, providing the ETArch controlling entity (i.e., the DTSAs) with context information about the link layer of both underlying network entities and MNs as well as with mechanisms to facilitate and optimize handover procedures. In doing so, ETArch can target a whole new set of network management scenarios, enabled and supported by the context information and mechanisms provided by IEEE 802.21. The specificities of the content being delivered, the number of users requesting the same content (i.e., subscribed to the same workspace), and the link conditions of wired and wireless networks are factors that can help to enhance content-reaching procedures ensuring that the content is sent to the subscribers through the most optimal path. For example, by leveraging context information of the link-layer, the DTSA becomes aware of the link conditions of MNs connected to the network under its domain, as well as neighbor PoAs detected by the MN, including their link parameters and conditions, which can then be used to control and optimize handover procedures of the MNs. Likewise, the DTSA can use this context information to (re)configure the flows across the whole network by using OpenFlow capabilities, aiming to optimize the usage of the network resources.

An overview of the proposed framework merging both ETArch and IEEE 802.21 mechanisms is presented in Figure 4.10.

The DTSA (i.e., the controlling entity) is the central entity responsible for the registration of DTS entities and management of DTS workspace. As such, it communicates with network entities via OpenFlow so it can (re)configure the DTS workspaces via *OFPT_FLOW_MOD* messages. As in the previous section, the proposed framework in this section also envisages the configuration of OpenFlow flows up to the network side endpoint, such as wireless network point of attachments (e.g. WLAN Access Point). This is required because DTS workspace configuration needs to be configured in every equipment in the path between endpoints, including the PoAs. Finally, DTS-enabled clients, which are represented in Figure 4.10 as the MNs, communicate via ETCP protocol with the DTSA.

With the extensions herein proposed, the DTSA can also communicate with PoAs

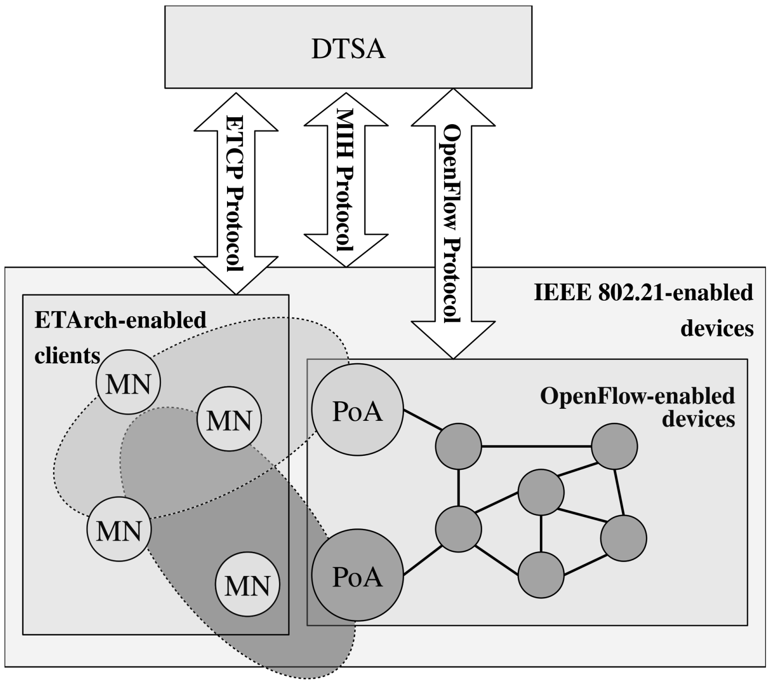


Figure 4.10: Proposed IEEE 802.21-enabled ETArch framework

and core network entities via the MIH protocol of the IEEE 802.21 standard, allowing it to acquire context information about their link layer. In addition, using the MIH protocol, the DTSA is able to communicate with MNs to also acquire context information about their link layer (e.g., current signal strength and detected PoAs) but it is also able to assist MNs in the handover procedures. The technology-abstraction of IEEE 802.21 enables the proposed framework to be applied to different access technologies, both wireless (such as, WLAN, LTE and WiMAX) and wired (such as, Ethernet) technologies.

With these extensions, the DTSA becomes able to dynamically and preemptively extend DTS workspaces towards the handover candidates (i.e., PoAs) detected by the MN while it moves, mitigating the effects of the handover procedure.

The proposed framework is composed by three main entities: (i) DTSA; (ii) EDOBRA switch (i.e., an edge or core switch); and (iii) MNs. Next, each of these entities is described individually.

4.3.1.1 Domain Title Service Agent

The DTSA is the controlling entity of the proposed framework, acting as an OpenFlow Controller for implementing the DTS workspaces in the network, and as a PoS for acquiring link-layer information of the entities under its domain and for assisting in the handover procedure of MNs. Its internal architecture is depicted in Figure 4.11, on which the dashed modules are the ones developed under the scope of this work and the remaining modules corresponding to the base implementation of the DTSA.

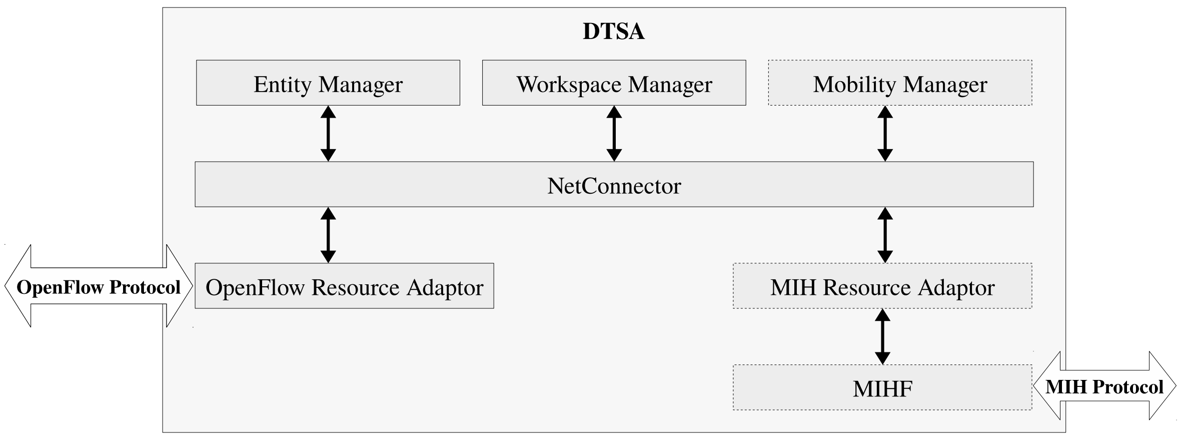


Figure 4.11: DTSA internal architecture

In what concerns its base operation, the DTSA is responsible for storing information about registered DTS entities (via *Entity Manager* module) and DTS workspaces (via *Workspace Manager* module), acting as an OpenFlow Controller to (re)configure the forwarding tables of the EDOBRA switches so that DTS workspaces are implemented in the underlying network. Regarding the extension to support mobility, it acts as an PoS (via *Mobility Manager* module) responsible for handling and controlling the mobility procedures of registered MNs, and to acquire context information from the link-layer of the network entities and MNs under its domain. These functions are interfaced by a central module (*NetConnector Manager* module) responsible for forwarding messages between the managers and the corresponding resource adaptors. The *OpenFlow Resource Adaptor* handles OpenFlow messages, while the *MIH Resource Adaptor* handles messages from the MIH protocol. Finally, it features a MIHF for exchanging MIH protocol messages with remote IEEE802.21-enabled entities, namely network entities (e.g., core switches and PoAs) and MNs.

4.3.1.2 EDOBRA Switch

The EDOBRA Switch (depicted in Figure 4.12) is an extension of a standard OpenFlow switch enhanced with IEEE 802.21 capabilities.

Besides storing information on how packets of each DTS workspace should be treated (in the form of flow entries) and executing data packet forwarding operations regarding its operation as an OpenFlow switch, the EDOBRA Switch uses IEEE 802.21 mechanisms for controlling and monitoring the link layer of its network interfaces. It incorporates a *Link SAP* for each network interface, which abstracts the upper layers on the specificities of each link technology in what concerns the control and monitor

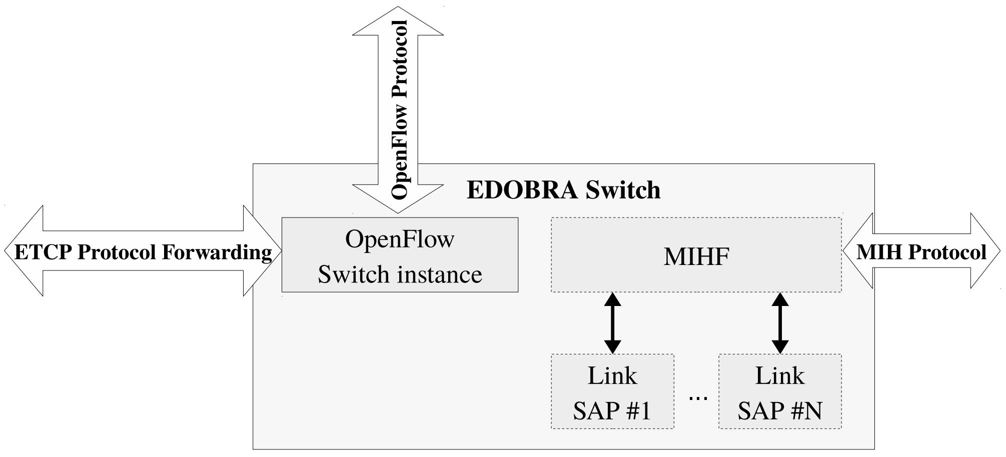


Figure 4.12: EDOBRA Switch internal architecture

operations of the corresponding interface. For example, whenever a new MN connects or disconnects from a given access network, the *Link SAP* of the corresponding interface generates an event towards entities that subscribed that event. The *Link SAPs* are locally interfaced by an *MIHF* that enables its interaction with remote entities, such as the MN and the DTSA. If the EDOBRA Switch is on the edge of the network, as part of its operation over the ETArch architecture, it is responsible for (de)encapsulating ETCP messages into OpenFlow messages. More specifically, incoming ETCP messages from DTS-enabled clients are encapsulated into *OFPT_PACKET_IN* messages and forwarded towards the DTSA, and ETCP messages originated in the DTSA are extracted from the *OFPT_PACKET_OUT* messages and forwarded towards the DTS-enabled clients.

4.3.1.3 Mobile Node

The MN (depicted in Figure 4.13) represents a mobile DTS-enabled client that establishes connection with the network.

The MN may be equipped with one or more access technologies, either wired (e.g., Ethernet) or wireless (e.g., WLAN or 3G). The MN is coupled with a *Mobility Manager* module responsible for handling and assisting the handover procedures. An MIHF is deployed in the MN, allowing higher-layer entities in the device itself (e.g., *Mobility Manager* module) or external network entities (e.g., DTSA) to control the links and to retrieve context information in an abstract way. Each *Link SAPs* is responsible for controlling and monitoring a specific network interface, abstracting the upper layers from the specificities of each link technology. Thus, the MN is able to e.g. retrieve the link conditions related with its current connection or provide information about other PoAs on its range. Regarding DTS procedures (e.g., entity registration, workspace creation and workspace (de)attachment), the MN contains a DTS Enabler (i.e., DTS

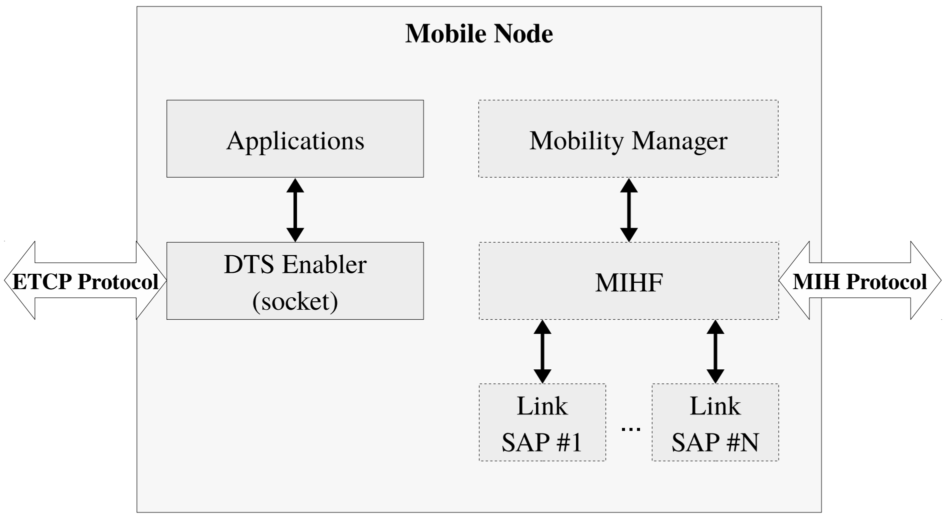


Figure 4.13: Mobile Node internal architecture

socket implementation) that allows it to exchange ETCP protocol messages with the DTSA and to send/receive messages to/from subscribed workspaces.

4.3.2 Handover Procedures Workflows

As in Section 4.2.3, a set of procedures are executed before and after the MN actually handovers to a new PoA, aiming to mitigate the effects caused by the handover delay. Therefore, with the purpose of supporting mobility in ETArch, the handover procedure is also divided into three main phases: (i) handover preparation; (ii) handover commit; and (iii) handover complete. On each phase, the DTSA, as the controlling entity of the network under its domain for both OpenFlow and IEEE 802.21 mechanisms, is responsible for controlling and optimizing the handover procedure of MNs. This includes both the verification of available resources on possible handover candidates and the reconfiguration of DTS workspaces to accommodate eventual changes that arise from the mobility of MNs.

4.3.2.1 Handover preparation

This phase aims at detecting possible candidates for the MN to handover to and order them in terms of preference based on several parameters. For that, the DTSA follows the algorithm presented in Algorithm 1).

In the handover preparation phase, the DTSA starts by identifying existing PoAs in range of the MN that are possible candidates for handover. This may require the DTSA to query the MN about PoAs in its range via IEEE 802.21 mechanisms. Then, for each available network in the range of the MN (step 2 in Algorithm 1), the DTSA checks the available resources with the purpose of verifying if it can accommodate the

Algorithm 1: Handover Preparation

```
1 Get all workspaces subscribed by the MN
2 for each network in the range of the MN do
3     Check available resources
4 Order networks by preference order
5 Return results to MN
```

attachment of the MN (e.g., how many clients are currently connected to the PoA and the average throughput in the PoA), as well as the impact that it could have in the core network (step 3 in Algorithm 1). This also includes the verification of which workspaces subscribed by the MN are currently being served by the candidate PoAs, so that the multicast nature of ETArch could be leveraged and, thus, optimizing the usage of network resources. For example, assuming that the mobile device MN_1 is subscribed to workspaces W_1 and W_2 and that a given PoA P_1 is already forwarding traffic related with W_1 towards its access network due to an existing subscriber, if MN_1 moves to PoA P_1 only the stream related with W_2 needs to be configured to be sent towards the MN via PoA P_1, since the stream of workspace W_1 can be shared with all subscribers (including the MN_1).

Based on all this information, the DTSA orders the candidate PoAs by the most to the least preferred handover target (step 4 in Algorithm 1), which are then sent to the MN (step 5 in Algorithm 1).

4.3.2.2 Handover Commit

Based on the list provided by the DTSA in the previous phase, the MN selects the target PoA (i.e., the PoA to which the MN intends to handover to). Several factors could be on the base of this decision. One of them is the detected received signal strength indicator (RSSI). The MN notifies the DTSA about its decision, which in turn starts the configuration of the network to accommodate the MN on its new location. More specifically, the DTSA initiates the procedure to preemptively extend the workspaces subscribed by the MN towards the target PoA, as described in Algorithm 2, before instructing the MN to handover to the target PoA. Such preemptive configuration of the workspaces has the purpose of reducing the time between the handover occurrence and the content reception in the MN after the handover.

The first step is for the DTSA to discover the tuple *(Switch ID, Port)* in the ETArch topology corresponding to the new location of the MN (i.e., target PoA) (step 1 in Algorithm 2). This tuple is discovered based on the MAC address of the target PoA. Then, the DTSA gets the workspaces subscribed by the MN (step 2 in Algorithm 2) and, for each one (step 3 in Algorithm 2), it associates the tuple *(Switch ID, Port)*

Algorithm 2: Handover Commit

```
1 Discover switch ID and Port based on target PoA MAC
2 Get all workspaces subscribed by the MN
3 for each workspace subscribed by the MN do
4     Associate (switch ID,Port) to the MN
5     Recalculate workspace path up to PoA
6     if PoA and Port not in workspace then
7         for each new switch in the path do
8             Configure workspace flow using OpenFlow
9 Instruct MN to handover to PoA
```

of the target PoA with the MN (step 4 in Algorithm 2). This associations means that the target PoA is interested in receiving the traffic related with the workspace, forwarding it via the port corresponding to the tuple. By doing this, the attachment to the workspace by the MN on the target PoA is being emulated. The workspace path is recalculated to take into consideration its extension towards the target PoA (step 5 in Algorithm 2). Then, if required (i.e., if the workspace is not yet implemented in the PoA) (step 6 in Algorithm 2), the DTSA implements the workspace via OpenFlow not only in the target PoA but also on intermediary switches to enable traffic from the workspace to reach the target PoA (step 7 and 8 in Algorithm 2).

At this point, traffic from the workspaces subscribed by the MN is being sent towards its current and target PoA. However, this implies that the network is momentarily sending data towards both current and new location of the MN. In doing so, as the MN handovers to the target PoA, it immediately starts receiving the traffic of the workspace. Otherwise, if the workspace is configured only after the handover procedure, additional delay occurs between the handover occurrence and the first received packet after the handover is introduced, namely the delay associated with DTS Entity registration and DTS workspace attachment.

After reconfiguring the network, the DTSA instructs the MN to handover to the target PoA (step 9 in Algorithm 2).

4.3.2.3 Handover Complete

The last phase starts after the L2 handover and it is related with the release of resources from the old location of the MN. It includes the reconfiguration of the network to remove, if required, workspace subscriptions associated with the MN from its previous PoA, as described in Algorithm 3.

As in the previous phase, the DTSA discovers the *(Switch ID, Port)* tuple in the ETArch topology corresponding to the previous PoA of the MN based on its MAC address. Then, for each subscribed workspace by the MN (step 1-2 in Algorithm 3),

Algorithm 3: Handover Complete

```
1 Get all workspaces subscribed by the MN
2 for each workspace subscribed by the MN do
3     Disassociate (switch ID,Port) from the MN
4     Recalculate workspace path
5     if path PoA removed from workspace then
6         for each superfluous switch in the path do
7             Remove workspace flow using OpenFlow
8 Report handover completion to MN
```

the tuple *(Switch ID, Port)* is disassociated from the MN (step 3 in Algorithm 3), after which the workspace path is recalculated (step 4 in Algorithm 3). Then, if no more subscribers to the workspace exist in the PoA, the DTSA, via OpenFlow, removes the corresponding flow entries related with the workspace from the flow tables of both the old PoA and intermediary switches in the path towards the PoA (step 5-7 in Algorithm 3).

Finally, the DTSA indicates to the MN that the handover has been terminated (step 8 in Algorithm 3).

4.3.3 Evaluation

In order to demonstrate the feasibility of the proposed solution and to evaluate its performance, the ETArch implementation was integrated with ODTONE[4] [28], an open-source implementation of IEEE 802.21 standard, according to the proposals in Section 4.3.1.

4.3.3.1 Scenario Description

The evaluation scenario, presented in Figure 4.14, was deployed over the a physical testbed in Brazil, under the scope of the FP7 OFELIA[5] project.

As depicted in Figure 4.14, two points of attachment (using *TP-Link TL-WR1043ND* equipment), namely PoA #1 and PoA #2, are providing two different access networks. The PoAs consists in EDOBRA switches and, therefore, they are enhanced with OpenFlow and ODTONE software, so that they become OpenFlow- and IEEE 802.21-enabled devices. Both PoAs are connected to a common OpenFlow Switch (using *Datacom DM4000 ETH24GX+2x10GX* equipment). The DTSA is connected to the OpenFlow devices (i.e., the PoAs and the OpenFlow Switch) using two different

[4]Open Dot Twenty ONE (ODTONE) - http://atnog.av.it.pt/odtone

[5]OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) - http://www.fp7-ofelia.eu

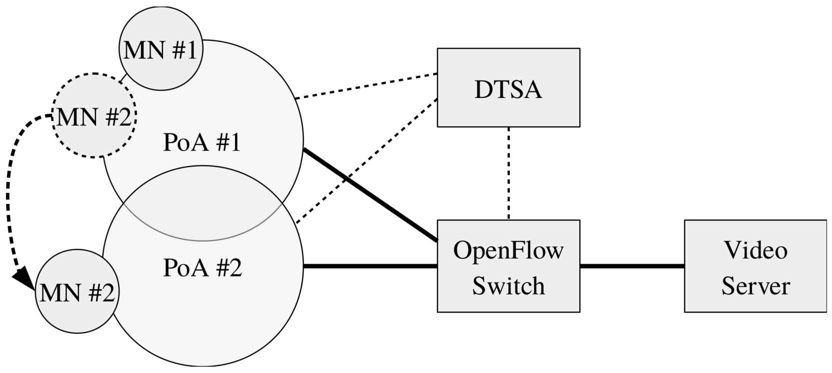


Figure 4.14: Scenario description

connections: one for control and another for data. The remaining entities that complete the evaluation scenario are two MNs, namely MN #1 and MN #2, initially connected to PoA #1, and a video server connected to the OpenFlow Switch. The video server is broadcasting a video over the "edobra" workspace and the two MNs are subscribed to that same workspace in order to receive the video stream. In addition, IEEE 802.21 messages are also sent over DTS protocol, being created for that purpose the "odtone" workspace.

In this scenario, MN #2 moves from PoA #1 to PoA #2, while receiving the video stream broadcasted by the video server. The MN #2 is initially attached to PoA #1 and already receiving the content, initiating the handover procedure to PoA #2 at time 10s of the experiment.

Each experiment was run 10 times and averaged results with a 95% confidence interval are here presented.

4.3.3.2 MN-initiated handover evaluation scenario

In this evaluation, the handover procedure is initiated by the MN, using the signalling presented in Figure 4.15. The MN #2, while on the move, detects a network with better signal strength, initiating the handover procedures required to move to the detected PoA (i.e., PoA #2).

Despite that it was chosen to trigger the handover based on the perceived signal quality by the MN, the proposed framework is flexible enough to accommodate other triggers for the handover procedure. For example, these could be based on the detection of a preferrable access technology or on the detection that the current link is having an high packet error rate. In addition, the flexibility of the proposed framework allows different entities to trigger the handover (i.e., MN-initiated or network-initiated handovers), likewise the ones described in Section 4.2.3. For example, the DTSA could trigger the handover of MNs subscribed to the same workspaces towards the same PoA,

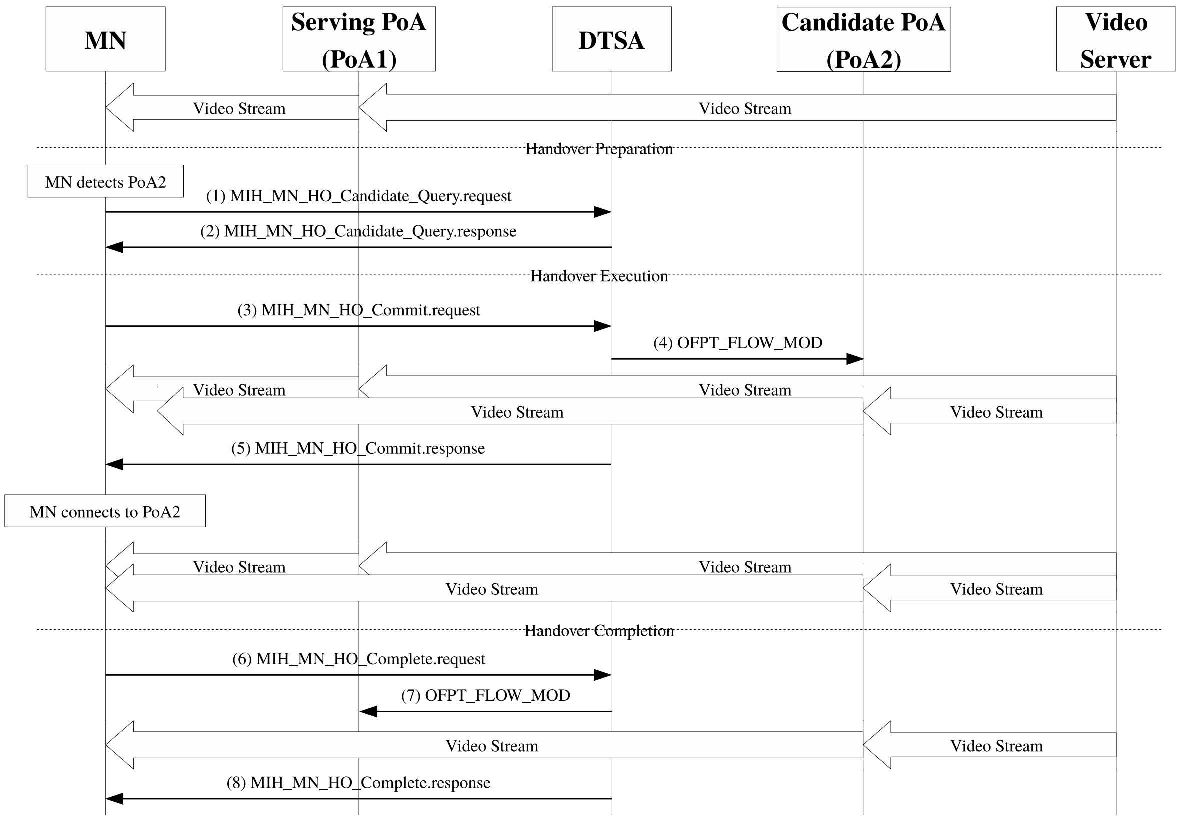

MN
Serving PoA (PoA1)
DTSA
Candidate PoA (PoA2)
Video Server
Video Stream
Handover Preparation
MN detects PoA2
(1) MIH_MN_HO_Candidate_Query.request
(2) MIH_MN_HO_Candidate_Query.response
Handover Execution
(3) MIH_MN_HO_Commit.request
(4) OFPT_FLOW_MOD
(5) MIH_MN_HO_Commit.response
MN connects to PoA2
Handover Completion
(6) MIH_MN_HO_Complete.request
(7) OFPT_FLOW_MOD
(8) MIH_MN_HO_Complete.response

Figure 4.15: MN-initiated handover scenario signaling

so that a more efficient use of the network resources could be achieved.

The evaluation scenario starts with the MN already registered in the DTSA and subscribed to the workspace of the video stream and, therefore, it is already receiving the video stream broadcasted by the video server. While on the move, the MN detects a new PoA with a better signal strength (i.e. the candidate PoA) than the one it is currently connected to. In doing so, and verifying at the same time that the signal of the current link has been losing strength, the MN issues a MN-initiated handover, using the MIH protocol, by sending a *MIH_MN_HO_Candidate_Query.request* message to towards the local DTSA (message 1 in Figure 4.15). The DTSA, which also encompasses the functions of PoS, verifies the resources available in the candidate PoA and if it is able to accommodate the MN's subscribed workspaces, as described in the Algorithm 1. The resources required along the whole path are also verified. The DTSA replies to the request of the MN by sending a a *MIH_MN_HO_Candidate_Query.response* message (message 2 in Figure 4.15).

The MN verifies if the candidate PoA is still feasible as a handover candidate. If so, it notifies the DTSA about the chosen target network by sending a *MIH_MN_HO_Commit.request* message (message 3 in Figure 4.15). Upon the reception of this message, the DTSA extends the "edobra" and "odtone" workspaces subscribed by the MN towards the candidate PoA, as described in the Algorithm 2. For that, the DTSA configures the flow tables of not only the candidate PoA and but also on all switches in the path between the candidate PoA and the video server, by sending *OFPT_FLOW_MOD* messages (message 4 in Figure 4.15). After the extension of both workspaces (i.e., "edobra" and "odtone" workspaces) towards the candidate PoA, the DTSA acknowledges the MN that the resources were reserved and that it can move to the candidate PoA (message 5 in Figure 4.15). At this point, the video stream is being duplicated to both the current and the candidate PoA access networks.

The MN executes the attachment to the candidate PoA and, since the workspaces were preemptively extended towards the new location of the MN, the MN immediately starts receiving the video stream. At this point, since a dual-interface MN is being considered, the MN is receiving the video stream by both network interfaces. After completing the handover procedure, the MN notifies the DTSA about its result by sending a *MIH_MN_HO_Complete.request* message (message 6 in Figure 4.15). Upon the reception of this message, the DTSA knows that it can release the resources allocated to the MN associated with its previous connection. This procedure, as described in Algorithm 3, includes the removal of the workspaces subscribed by the MN from the previous PoA, if no other subscribers to the same workspaces are connected to PoA #1 (message 7 in Figure 4.15). Finally, the DTSA acknowledges the MN about the conclusion of the handover procedures (message 8 in Figure 4.15).

4.3.3.3 Performance Evaluation

In this section, the performance of the proposed framework is evaluated and compared with a deployment scenario of ETArch without the mobility enhancements herein proposed. The results related with the impact on the content reception are shown in Table 4.3, presenting values on the duration and number of duplicated packets, lost packets during the handover and the time required to restore the stream after the handover. Figure 4.16 presents the content reception on the MN #2 using each solution, focusing on the time that the handover occurs.

Table 4.3: Content reception performance comparison

	ETArch with IEEE 802.21	**ETArch only**
Lost packets during HO	0	19.4 ± 1.0
Restore stream delay (ms)	0	407.9 ± 26.9
Redundancy at MN (ms)	40.3 ± 12.4	0
Redundancy at network (ms)	851.7 ± 32.1	0
Redundancy at network (packets)	39.6 ± 3.1	0

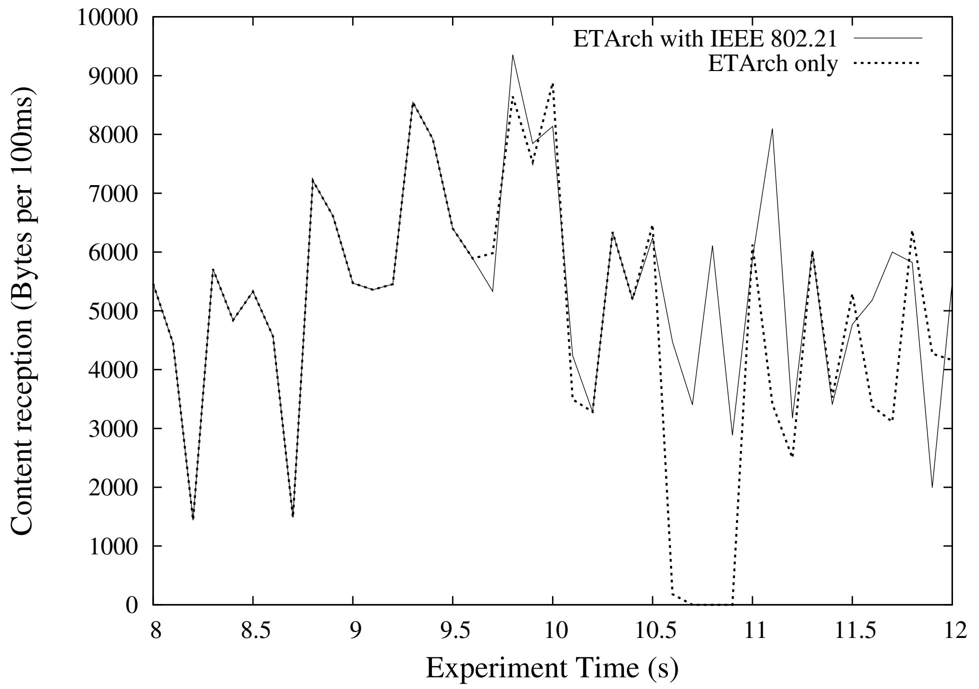


Figure 4.16: Content reception on the MN

From the results in Table 4.3, it can be verified that the proposed framework enabled the handover to occur with no packet loss, since a make-before-break approach was used. As such, the MN #2 only disconnected from PoA #1 when the connection with PoA #2 was already established. Thus, during the handover procedure, the MN temporarily received the video stream from both interfaces simultaneously, allowing its reception without interruptions (as observed in Figure 4.16). More specifically, the video stream was received simultaneously via both interfaces during approximately 40ms. Also, since workspaces were previously extended towards the candidate PoA before the handover itself, it resulted in no delay to restore the video stream. However, duplicated packets were sent towards the previous and the next location of the MN #2 for about 851.7 ± 32.1 ms, even if no subscribers were present, resulting in a waste of network resources.

Using the vanilla ETArch (i.e., without the IEEE 802.21 enhancements of the proposed architecture), the DTS video application on MN #2 needed to detach from the workspace and unregister from the DTSA through the old PoA (i.e. PoA #1) and to re-register and reattach to the workspace through the new PoA (i.e., PoA #2). This procedure took 407.9 ± 26.9 ms. During this time the MN was not subscribed to the workspace of the video stream and, therefore, it did not receive any packet from the video stream (as observed in Figure 4.16), representing approximately 19.4 lost packets. Moreover, in consequence of this behavior, there was no redundancy during the handover, in opposite to what occurs on the proposed framework.

Comparing the proposed framework with this solution, it can be verified that the proposed framework was able to prevent the packet loss, by extending the workspaces to the new PoA before the handover itself occurs, allowing it to receive the contents from both interfaces while moving across different networks. With the proposed framework, the handover becomes transparent to the DTS application, since the DTSA was able to extend the MN's workspaces towards its new location without requiring the DTS video application to perform entity registration and workspace attachment operations to restore the workspace flow.

4.3.3.4 Control Signaling Overhead Analysis

The results on the footprint of each protocol in the proposed handover signaling are presented in Table 4.4, showcasing the amount of data exchanged for each protocol and the total time required for the different phases of the handover procedures. The values for the amount of data exchanged consider the size of the whole message.

Table 4.4: Total signaling overhead per handover

		Handover Preparation	**Handover Commit**	**Handover Complete**
Size (bytes)	**IEEE 802.21**	235	171	218
	OpenFlow	0	2568	0
	ETCP	0	0	0
Time (ms)		108.3 ± 18.5	65.7 ± 17.5	32.0 ± 10.9

Analyzing the results from Table 4.4, almost 20% of the exchanged signaling corresponds to IEEE 802.21 protocol, with the remaining 80% related with OpenFlow protocol. While the OpenFlow signaling depends on the number of workspaces subscribed by the MN, the IEEE 802.21 signaling does not depend on it. As such, the percentage of IEEE 802.21 overhead could be even lower if the MN is subscribed to more workspaces. Regarding the ETCP protocol, no control signaling was required during the handover procedure since ETCP operations were handled internally by the DTSA and triggered by the IEEE 802.21 signaling. In terms of involvement of the MN #2, it accounted for about 20% of the total signaling overhead, due to its assistance in all phases of the handover process. Likewise, since the MN only handles MIH protocol messages, if it is subscribed to more workspaces, this value will decrease.

The handover commit was the phase on which more control signaling was exchanged. This occurred mainly because of the reconfiguration of the workspaces on each OpenFlow device, corresponding to approximately 86% of the total exchanged signaling. In this specific scenario, two workspaces (the video stream and the IEEE

802.21 control workspaces) required the reconfiguration towards the new PoA and, thus, each workspace that the MN was subscribed to increased the amount of OpenFlow signaling in about 1284 bytes.

Finally, in the handover complete phase, since another MN (i.e. MN #1) was attached to PoA #1 and subscribed to the same workspaces as the MN #2, the workspaces were not removed from PoA #1 in the handover complete phase. If the MN #2 was the only subscriber to the workspaces on PoA #1, the DTSA would have triggered the mechanisms to remove the workspaces from PoA #1, increasing the overhead introduced by the OpenFlow protocol (about 1268 bytes per workspace).

In terms of time impact, the handover preparation procedures are the most demanding phase (about 108.3 ± 18.5 ms). It encompasses the time to compute the available resources on each available PoA, with the purpose of finding the best handover candidate for the MN. In the evaluation scenario, the commit procedures required more time than the complete procedures because, since the MN #1 remained at PoA #1, there was no reconfiguration of the workspaces on PoA #1 after the handover. Still, the overhead in terms of time introduced by the IEEE 802.21 procedures accounts only to 21% of the total handover time. The remaining 79% respects to the L2 handover procedure, which took 779.9 ± 29.3 ms. The L2 handover procedure includes the scanning, association and authentication steps. Potential optimizations could be achieved with different L2 management possibilities.

4.3.4 Discussion

The multicast nature of workspaces in the ETArch architecture allows a many-to-many communication channel where messages sent by one entity in a given workspace are delivered to any other entity subscribed to that same workspace. However, since the base specification of ETArch does not guarantee reliability in the message delivery, it is highly important to mitigate the effects of the handover procedure of MNs (i.e., reduce the handover delay) or otherwise the moving entity may lose information during the handover procedure.

The mechanisms provided by IEEE 802.21 to acquire link-layer context information and to handle the handover procedure of MNs play an important role to facilitate and optimize the handover procedure, mitigating its effects. As such, by integrating the IEEE 802.21 mechanisms in the ETArch architecture, the DTSA becomes capable of handling the different phases of the handover procedure (i.e., preparation, commit and complete phases) of registered MNs and of reconfiguring the network according to their movement. As an example, the handover can be triggered by the movement of a given MN, with the DTSA accommodating the required changes to extend the workspaces towards the new location of the MN (i.e., reconfigures the implementation of the workspaces in the network forwarding equipment) before the handover actually occurs and prune the workspace, if required, after the handover is complete. This preemptive approach in reconfiguring the workspaces, on which the DTSA acts on behalf of the MN to subscribe the workspaces from its next location (i.e., PoA), enables the MN to immediately receive traffic from the subscribed workspaces after the handover is complete. Otherwise, the MN, after the handover procedure, would need to re-express its interest (i.e., resubscribe) on the workspaces, increasing the time to restore the communications. Such behavior can be observed from the obtained results of the evaluation of the proposed mobility framework for ETArch, where the preemptive approach allowed the MN to move to a new location without losing any message during the handover procedure, unlike to a native ETArch approach where message loss was verified.

Additionally, by having an overview of the network topology and context information of the wireless surroundings of both MNs and network entities, the DTSA also becomes capable of triggering the handover of a set of MNs subscribed to the same workspace to the same PoA with the purpose of leveraging the multicast capabilities of ETArch architecture and, consequently, optimizing the usage of network resources.

Although the proposed solution was exemplified using ETArch, these mechanisms are generic and could be adapted towards different ICN instantiations. For example, if applied to NDN architecture, the IEEE 802.21 procedures could be integrated to trigger the reconfiguration of the PIT and FIB of NDN routers in order to accommodate the required changes due to the MN movement.

4.4 Supporting Mobility Across Different Network Architectures

The realization of Future Internet network architectures is likely to occur in an incremental way over time, which will lead to a coexistence period of these novel network architectures in parallel with the current Internet architecture [39]. In particular, the initial roll out of ICN network architectures can occur on the edge of the network [88], by means of isolated network architectural islands interconnected with one another either through dedicated links or as an overlay over IP. This deployment approach creates a heterogeneous networking environment at the access networks, paving the way for a new set of possibilities regarding the mobility of MNs. MNs becomes capable of not only moving between access networks supporting the same network architecture, but also between IP-based and ICN-based access networks and between different ICN-based access networks as well.

While on the move across different network architectures, a given MN needs to be able to preserve its reachability to resources being accessed, even when the MN attaches to a different network architecture than the one where the resource is deployed in. This means that inter-network architecture mobility management mechanisms are required, allowing the MN to acquire information on how to reach resources from each network architecture and, consequently, allowing the MN to reestablish the access to the resource. In addition, whenever the MN and the resources are on different network architectures, interoperability mechanisms may also be required to convert messages across protocols. Otherwise, MNs are not able to access the resources when moving to a different network architecture, causing segmentation on the resource provisioning.

In this section, an inter-network architecture mobility framework is proposed, which enables MNs to move between access networks supporting different network architectures without losing its reachability to a given resource. This framework leverages the interoperability mechanisms defined in Chapter 3 to make resources available in a different network architectures, enhancing its operation to support inter-network architecture mobility upon request by the MNs. The mechanisms herein proposed focus on inter-network architecture mobility targeting IP and ICN network architectures in content retrieval use case scenarios. The proposed solution is generic and agnostic to the specific mobility management mechanisms of each network architecture, which still apply when the MN is moving on the same network architectural environment.

4.4.1 Motivational Scenario

The motivational scenario for this work (depicted in Figure 4.17) considers the mobility of Alice's device across access networks supporting different network architectures while downloading a given content from a remote content source.

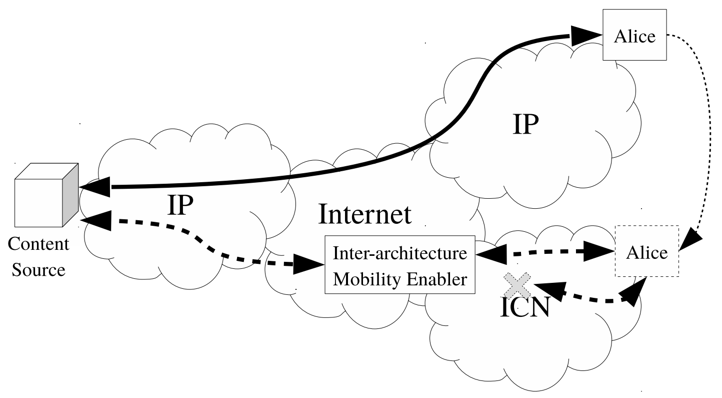


Figure 4.17: Reference motivational scenario

In particular, Alice is connected to the IP network architecture downloading a given content from a content source via the HTTP protocol. Since Alice and the content source are connected to the same network architecture, direct communication between both communication endpoints can be established. While moving across IP-based access networks, mobility is managed through any number of IP-based mobility management protocols (as discussed in Section 2.4.1). However, when Alice arrives home, her device automatically connects to the access network broadcasting therein, which operates solely based on the ICN protocol.

Because of this change of network architecture, Alice is unable to continue downloading the content. In this example scenario, such behavior is motivated by incompatibilities between the IP and the ICN architectures, and, therefore, messages belonging to the IP protocol are unable to be natively sent through the ICN network architecture (and vice-versa), being dropped.

In order to enable Alice to continue downloading the content, inter-network architecture mobility mechanisms are required to be put in place, allowing the device to maintain its reachability to content whenever it is initiated in a given network architecture and continued in a different network architecture. Alice's device can then, after the handover procedure, request the remaining of the content via the ICN protocol towards an intermediary entity which is responsible for acquiring the content on its behalf via HTTP (over IP) protocol.

4.4.2 Proposed framework

The proposed framework (shown in Figure 4.18) introduces a new entity in the network, named Future Internet eXchange Anchor (FIXA), which acts as an anchor point and as a gateway whenever a MN moves to a different network architecture than the one where the content is physically deployed in (i.e., whenever the communication endpoint moves between different network architectures). While moving across access networks of the same network architecture, the existing mobility management mechanisms on each network architecture are used for handling the mobility of the MN.

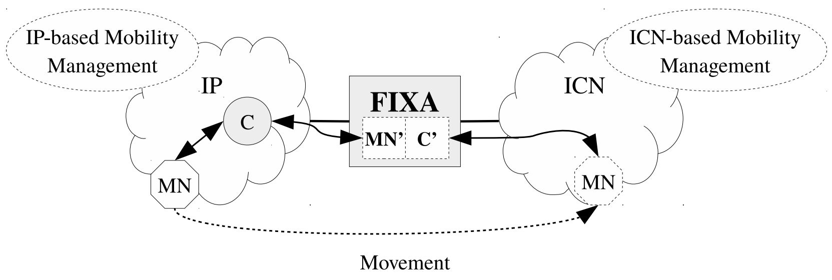


Figure 4.18: Framework Overview

If the MN and the content source are deployed in the same network architecture, both endpoints communicate directly with one another, not requiring the FIXA to mediate the communication. However, when the MN moves to a different network architecture, the content may become unreachable through the protocols previously used by the MN. In order to maintain its reachability to the content, the MN requests the FIXA to provide an address to the content compatible with its current network architecture, enabling the content to be reached through the FIXA.

In the example of Figure 4.18, the content C, deployed in IP, remains reachable by the MN when it moves to ICN, through the FIXA that intermediates the communication between the MN and the content. Note that from the *IP-based Mobility Management* point of view, the MN simply moved to a new AP, the FIXA (and remains there as long as the MN roams on the ICN). While the MN moves on the ICN-based networks, the mobility mechanisms previously proposed in this chapter can be used.

4.4.2.1 Future Internet eXchange Anchor (FIXA)

The Future Internet eXchange Anchor (FIXA) is introduced in the network by the proposed framework, being the core enabler for providing an inter-network architecture mobility management service. In other words, this entity is responsible for enabling content in a given network architectures to still be reached by the MN when it moves to a different network architecture. For that, the FIXA not only acts as an anchor point for the MN for communications across different network architectures but also acts as a gateway that converts messages between the protocols supported by each network architecture. It is natural that the FIXP (as proposed in Chapter 3) could be a possible entity for the deployment of the FIXA functionalities, although it is not mandatory and the FIXA could be deployed as a standalone entity.

The inter-network architecture mobility mechanism is initiated by the MN which requests the FIXA to generate an on-demand mapping for the content being accessed. This mapping defines how the content can be accessed from a different network architecture than the one where it is physically deployed in. The mapping needs to comply with two main properties: (i) uniquely identify a content in the network architecture where it is deployed in; and (ii) enable messages to be forwarded towards the FIXA.

When the MN addresses the generated mapping, the FIXA anchors the communication, allowing messages destined to content deployed in a different network architecture than the MN to be sent towards the FIXA. Additionally, since the protocols may not be compatible across the different network architectures, the FIXA intermediates the communication between the MN and the content source, acting as a gateway that converts messages between the supported protocols. In the process, the FIXA needs to behave simultaneously as the destination and source of messages, establishing two different connections: one between the MN and the FIXA (acting on the behalf of the content source) and another between the FIXA (acting on the behalf of the MN) and the content source. Requests received by the FIXA from the MN are converted and sent to the correspondent content source. Similarly, responses from the content source are received by the FIXA, which are then converted and sent to the MN.

4.4.2.2 Mobile Node (MN)

The MN is also target of enhancements for supporting the proposed inter-network architecture mobility mechanisms. Figure 4.19 presents the enhanced internal architecture of a MN supporting both IP and NDN network architectures in a dual stack operation. For simplicity, the discussion and implementation will use the NDN architecture, given its implementation maturity. Nevertheless, the proposed solution is generic and flexible to be extended and to adapt to other ICN architectures.

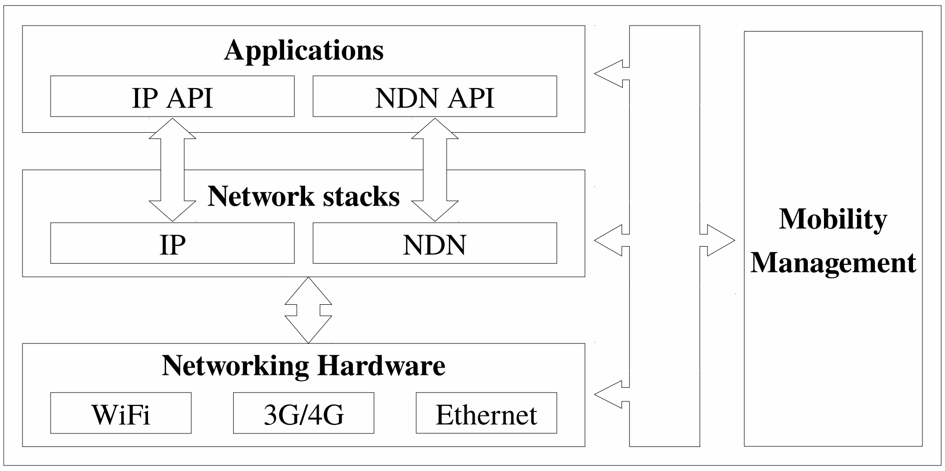


Figure 4.19: Enhanced internal architecture of MN

It is composed by three layers responsible for the operations related with the applications, network stacks and networking hardware and by a cross-layer responsible for operations related with the inter-network architecture mobility procedures.

Application Layer: Applications willing to support inter-network architecture mobility need to implement the logic to access content using the protocols supported on each network architecture. In the example of Figure 4.19, this means that the application needs to implement the procedure to acquire the content via protocols supported in the IP network architecture (e.g., using HTTP protocol) and in the NDN network architecture (e.g., using the NDN protocol). Applications can then send and receive messages when connected to different network architectures.

Network Stacks Layer: The MN itself is enhanced with the support of multiple networking stacks in a dual stack operation. In the example of Figure 4.19, the MN simultaneously supports both IP and NDN networking stacks. The MN can then be able to receive, identify, process and send messages related with each supported network architecture.

Networking Hardware Layer: For remote communications, the MN may encompass multiple network interface cards (NICs) of the same or different link technologies, either wired (e.g., Ethernet) or wireless (e.g., WiFi and 3G/4G).

Mobility Management Layer: The mobility management layer is responsible for managing the mobility procedures on the MN, including management of the inter-network architecture mobility procedures. It consists of a cross layer component that interacts with the remaining layers (i.e., networking hardware, network stacks and applications) as follows:

- detect changes regarding the connections of the MN and the PoAs;
- discover the network architectures supported by the access networks that the MN is connected to;
- notify applications about changes regarding the supported network architectures.

This layer is also responsible for handling the local mobility of the MN on each network architecture, using specific mobility management mechanisms existing therein.

4.4.2.3 Resource Binding Procedure

In order to allow applications in the MN to continue reaching the content after the handover of the MN to a different network architecture, a discovery procedure - the *Resource Binding Procedure* (RBP) - is defined by the proposed framework. This procedure allows the applications in the MN to discover how to reach the content being accessed before the handover from the new network architecture, so that it can reestablish the download of the content.

The RBP (Figure 4.20) is established between the MN and the FIXA, allowing the MN to request the FIXA to provide access to the content using protocols supported in the new network architecture.

When the MN moves to a different network architecture, it issues a *Resource Binding Update* (RBU) message for each content being accessed before the handover. This message includes information about the address of the content on the network architecture where it is deployed in (i.e., the original URI), as well as optional metadata (e.g. the preferred protocol to be used in the new network architecture). Upon

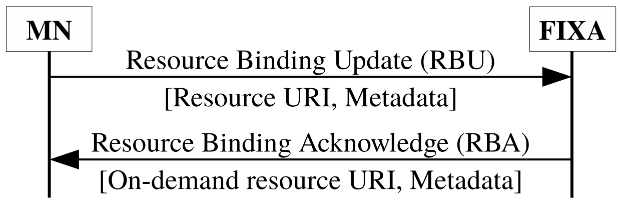

MN
FIXA
Resource Binding Update (RBU)
[Resource URI, Metadata]
Resource Binding Acknowledge (RBA)
[On-demand resource URI, Metadata]

Figure 4.20: *Resource Binding Procedure*

reception of the RBU message, the FIXA generates an on-demand mapping to the content identified in the RBU, which is then delivered to the MN via *Resource Binding Acknowledge* (RBA) message, along with metadata related to the protocol used to access the content from the new network architecture. By addressing the mapping acquired via the RBP procedure, the application in the MN can reach the content from the new network architecture, but intermediated by the FIXA.

4.4.3 Evaluation

To verify the practicality of the proposed framework in providing reachability to content after the handover of the MN to a different network architecture, the framework was implemented and deployed over the evaluation scenario depicted in Figure 4.21, resorting to NDN implementation as the ICN instantiation. For simplicity and in order to focus on the inter-network architecture mobility mechanisms, it was not considered the mobility management inside each network architecture.

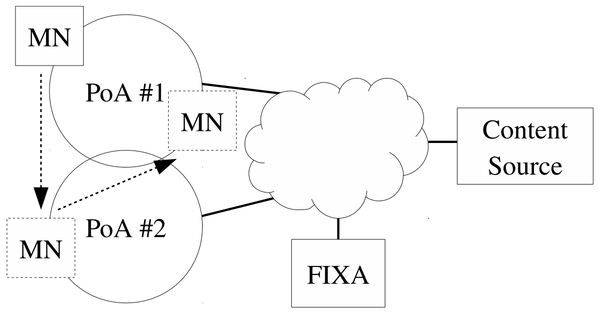


Figure 4.21: Evaluation Scenario

This scenario is composed by two PoAs supporting IP or NDN architectures, a FIXA, a content source supporting only IP or NDN network architectures and a single interfaced MN supporting IP and NDN architectures. NDN (NDN Platform v0.6.2[6]) is deployed as an overlay over the IP network. Notwithstanding, if the MN is connected to the PoA supporting NDN, the application only communicates via the NDN protocol.

The MN and the PoAs are physical machines configured with an AMD Embedded G series GX-412TC processor and with 4GB RAM running Ubuntu 14.04, while the

[6]NDN Platform - http://named-data.net

content source and the FIXA are deployed in virtual machines with 8 core processor and 8GB RAM running Ubuntu 16.04.

The evaluation of the proposed framework focused on verifying that the application is able to correctly download the content while the MN moves across different network architectures, as well as assessing the impact of the FIXA in the communication between the MN and the content source.

The experiments were run 30 times and the averaged results with a 95% confidence interval are presented.

4.4.3.1 Use Case Signaling

Two different use case scenarios were evaluated for the handover of a given MN between IP and NDN, differing on the network architecture where the content source was deployed in and the network architecture that the MN was initially connected to.

Use case 1: From IP to NDN to IP In this use case scenario (Figure 4.22), the content source is deployed in IP network architectures, providing access to content via HTTP protocol.

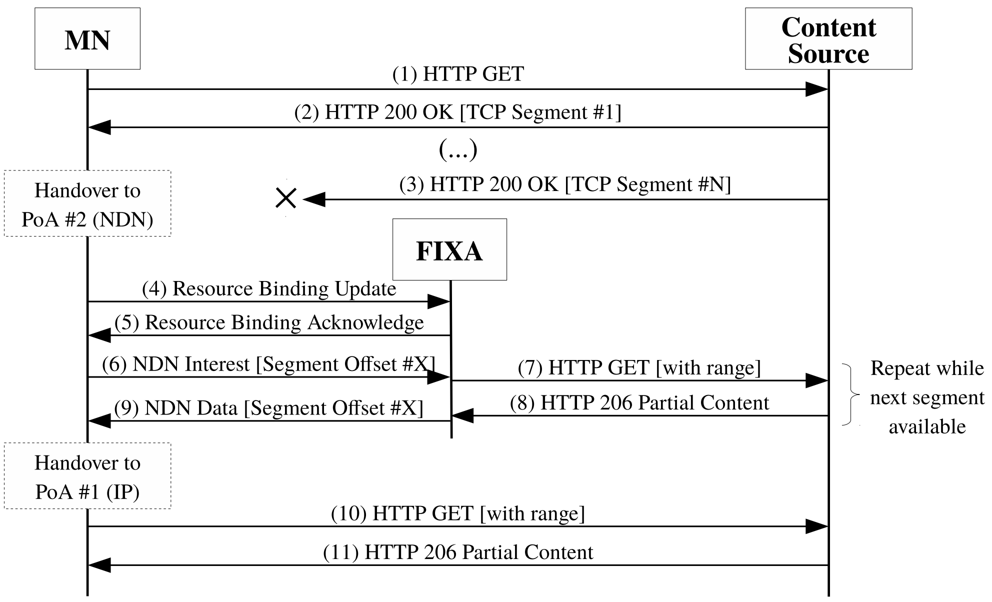

MN
Content Source
(1) HTTP GET
(2) HTTP 200 OK [TCP Segment #1]
(...)
Handover to PoA #2 (NDN)
(3) HTTP 200 OK [TCP Segment #N]
FIXA
(4) Resource Binding Update
(5) Resource Binding Acknowledge
(6) NDN Interest [Segment Offset #X]
(7) HTTP GET [with range]
(8) HTTP 206 Partial Content
(9) NDN Data [Segment Offset #X]
Repeat while next segment available
Handover to PoA #1 (IP)
(10) HTTP GET [with range]
(11) HTTP 206 Partial Content

Figure 4.22: From IP to NDN to IP

The MN is initially connected to an IP-based access network (i.e., PoA #1) and, since the content source is deployed over the same network architecture (i.e., IP), the MN starts downloading the content from the content source via HTTP (messages 1 and 2 in Figure 4.22).

After moving to the PoA #2, which only supports the NDN network architecture, the MN cannot directly download the content from the content source because they are on different network architectures (message 3 in Figure 4.22). The MN initiates the *Resource Binding Procedure* with the purpose of discovering how to reach the content from the NDN network architecture (messages 4 and 5 in Figure 4.22). The discovered mapping allows the MN to reestablish the communication with the content source, intermediated by the FIXA which converts messages from one protocol into the other (i.e., *NDN Interests* into *HTTP GET* messages and *HTTP response* messages into *NDN Data* messages). To continue the download, a *NDN Interest* message addressing the next byte segment offset is issued (message 6 in Figure 4.22). The byte segment offset is used by the FIXA to include the *Range* header in the *HTTP GET* message (message 7 in Figure 4.22) so that only the specific part of the desired content is downloaded, avoiding the need to download the entire content from the content source. After receiving the *HTTP 206 Partial Content* message (message 8 in Figure 4.22), the FIXA extracts the content from the received message, which is then used to generate the corresponding *NDN Data* to be sent to the MN (message 9 in Figure 4.22). If an HTTP error message is received by the FIXA, it returns a *NDN Data* message to the MN containing a negative acknowledgement (NACK).

Upon moving back to the PoA #1, the MN requests the remaining content directly from the content source (messages 10 and 11 in Figure 4.22). In this case, since part of the content has already been received, the MN includes the *Range* header in the *HTTP GET* message in order to request only the missing content.

Use case 2: From NDN to IP to NDN In this use case scenario (Figure 4.23), the content source is deployed in NDN network architectures, providing access to contents via NDN protocol.

The MN is initially connected to PoA #1 which supports the NDN network architecture and, since the content source is also deployed over the NDN network architecture, the MN requests the content from the content source via the NDN protocol (messages 1 and 2 in Figure 4.23).

After moving to an access network supporting only IP (i.e., PoA #2), the MN is not able to directly get the content from the content source, due to being on different network architectures (messages 3 and 4 in Figure 4.23). As such, it triggers the *Resource Binding Procedure* to discover how to reach the content from the IP network architecture (messages 5 and 6 in Figure 4.23). The MN can then request the remaining content through the FIXA, which converts messages between network architectures. More specifically, it converts *HTTP GET* messages into *NDN Interests* and *NDN Data* messages into *HTTP response* messages. Based on the *Range* header of the *HTTP GET* messages (message 7 in Figure 4.23), the FIXA issues the *NDN Interest* message (message 8 in Figure 4.23) for the corresponding byte segment offset.

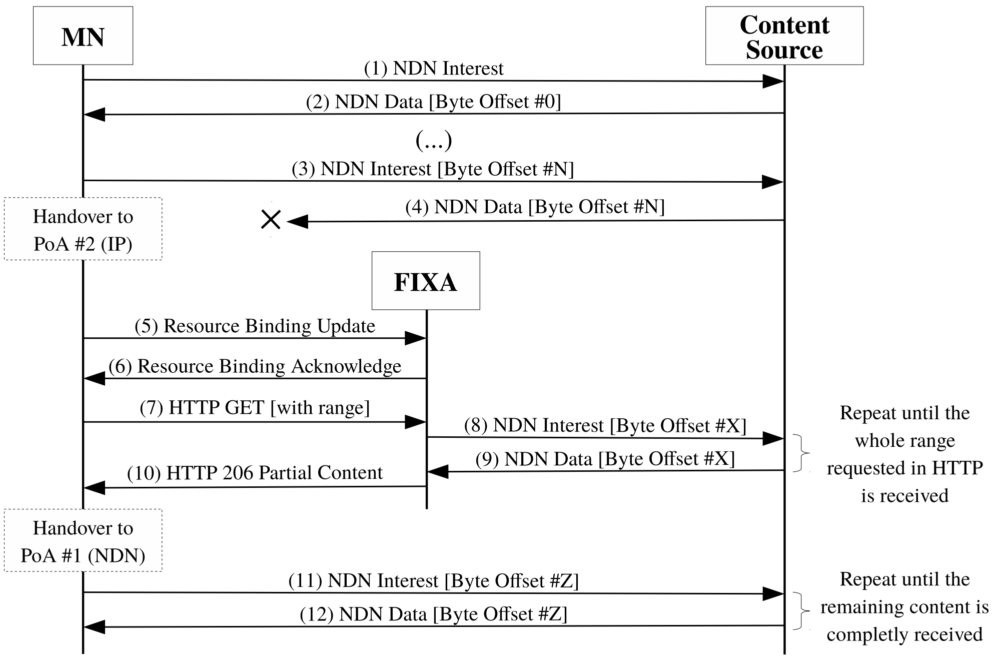

MN
Content Source
(1) NDN Interest
(2) NDN Data [Byte Offset #0]
(...)
(3) NDN Interest [Byte Offset #N]
Handover to PoA #2 (IP)
(4) NDN Data [Byte Offset #N]
FIXA
(5) Resource Binding Update
(6) Resource Binding Acknowledge
(7) HTTP GET [with range]
(8) NDN Interest [Byte Offset #X]
(9) NDN Data [Byte Offset #X]
(10) HTTP 206 Partial Content
Repeat until the whole range requested in HTTP is received
Handover to PoA #1 (NDN)
(11) NDN Interest [Byte Offset #Z]
(12) NDN Data [Byte Offset #Z]
Repeat until the remaining content is completly received

Figure 4.23: From NDN to IP to NDN

Whenever the requested content is received by the FIXA (message 9 in Figure 4.23), it generates a *HTTP 206 Partial Content* message to be sent to the MN (message 10 in Figure 4.23). If a *NDN Data* message containing a NACK is received by the FIXA, an HTTP response message with an error code is sent.

When the MN handovers back to PoA #1, it requests the remaining content via the NDN network architecture (messages 11 and 12 in Figure 4.23), issuing *NDN Interest* messages for the next byte offset segment.

4.4.3.2 Validation

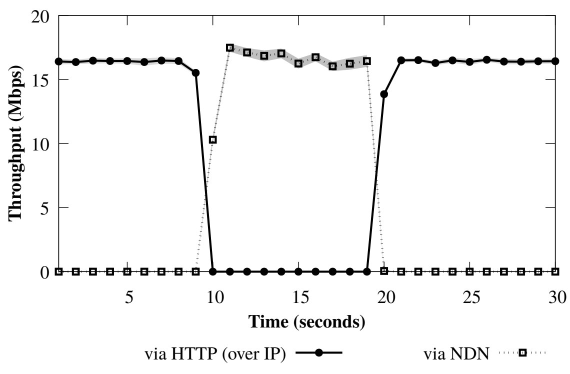
To validate the mobility mechanism of the proposed framework, the use cases depicted above were implemented and deployed over the evaluation scenario. As such, while downloading the content, the MN handovers between both IP-based and NDN-based access networks. In doing so, the application downloaded part of the content via HTTP (over IP) and NDN protocols. After completing the download of the content, the SHA256 hash of the downloaded content by the application was generated and compared with the SHA256 hash of the correspondent content in the content source, verifying that both hashes match. This is an indicator that the content was correctly downloaded by the application while the MN moved across different network architectures.

4.4.3.3 Performance Evaluation

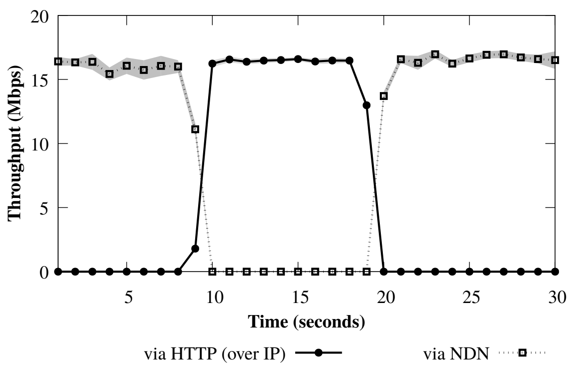
In the evaluated use cases, the performance in NDN in terms of throughput was affected by two different parameters: (i) the Interest window size; and (ii) the RTT between the MN and the content source. The main reason is related with the consumer-driven approach taken by NDN, where an *NDN Data* message is only delivered to the consumer upon the issue of the correspondent *NDN Interest* message. For example, by using an Interest window of size 1, a new *NDN Interest* message is only issued after the previous *NDN Interest* message is satisfied (i.e., after receiving the corresponding *NDN Data* message). Consequently, the time between *NDN Interest* messages is affected by the RTT between the MN and the content source.

To improve the throughput capability of NDN in the evaluated use cases, NDN was configured to use a Interest window of size 5 to allow simultaneous requests of different parts of the content and the *SignatureSha256WithEcdsa* algorithm was used to reduce the delay in signing the *NDN Data* messages and, consequently, the RTT between the MN and the content source.

The throughput measured in the MN regarding the download of a given content while moving across IP-based and NDN-based access networks is presented in Figure 4.24.



(a) IP-NDN-IP use case



(b) NDN-IP-NDN use case

Figure 4.24: Performance in terms of throughput measured in the MN

As can be observed, in both use case scenarios, the application in the MN is able to resume the download after moving to a different network architecture. Moreover, after moving to PoA #2 (meaning that both the MN and content source are in different network architectures and, consequently, the communication needs to be intermediated by the FIXA), a similar performance in terms of throughput was achieved when compared with the case where the MN and the content source are on the same network architecture and no intervention of the FIXA is required.

4.4.3.4 Resource Binding Procedure Delay

After moving to a different network architecture than the one where the content source is physically deployed in, the MN triggers the *Resource Binding Procedure* to discover the corresponding mapping to the content being accessed in order to be able to continue reaching it from its current network architecture. This procedure took about 2,83 ± 0,36 ms when requested over NDN and 1,10 ± 0,19 ms when requested over HTTP in the IP network architecture.

This procedure is more time consuming in NDN than in IP mainly because of two reasons: (i) in NDN, the FIXA needs to register the generated mapping with the connected NDN forwarder; and (ii) since the RBA message is piggybacked in an *NDN Data* message, additional delay is introduced due to the need to sign the *NDN Data* message.

Nevertheless, independently of the network architecture where the RBP procedure takes place, the delay is only introduced once in the communication and it impacts the time to restore the download of the content after moving to a different network architecture.

4.4.3.5 FIXA Delay

Figure 4.25 depicts the delay introduced by the FIXA, regarding its capability of acting as a gateway in the communication.

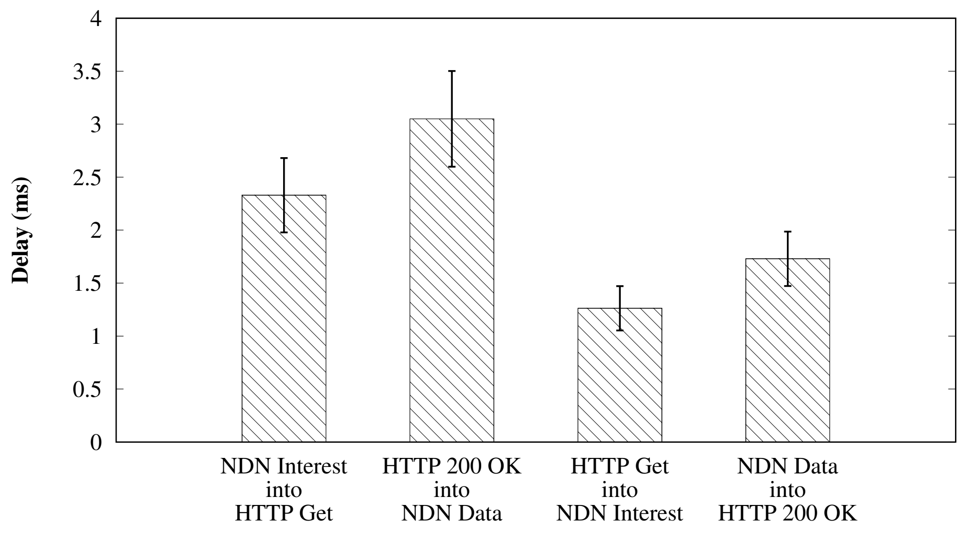


Figure 4.25: Delay introduced by the FIXA

The delay introduced by the FIXA includes handling the messages while acting as a source and destination of the messages on each network architectured, as well as, processing related with the conversion of messages from one protocol into another. In our implementation of the proposed framework, the FIXA introduced a delay between 1,2 ms and 3,1 ms, increasing the RTT in the communication between the MN and the content source.

4.4.4 Discussion

With the maturation of the novel network architectures proposed in the recent years, their incremental deployment along with the current IP network architecture is a highly potential scenario. A possible location for their initial deployment is at the edge of the network, targeting specific use case scenarios, by means of isolated islands interconnected with one another as an overlay over IP or using dedicated links. As such, mobility of a given entity between access networks supporting different network architectures becomes an added possibility.

To tackle such scenario, both interoperability and mobility concepts were integrated in a single mobility framework, allowing novel mobility scenarios to be taken into consideration in heterogeneous environments supporting IP and ICN network architectures. In particular, the MN becomes capable of maintaining its reachability to a given content when the former moves from an IP-based to an ICN-based access network (and vice-versa). The proposed mobility mechanism complements the local mobility management mechanisms that exist on each network architecture (as those previously proposed in this thesis) which aims to provide reachability/connectivity inside the same network architecture. Nonetheless, when the MN moves to a different network architecture, the handling FIXA is seen by the local mobility management mechanisms as the new point of attachment of the MN.

This framework was implemented and evaluated in content retrieval use case scenarios, exemplified using both IP and NDN network architectures. Results allowed to validate the correct download of a given content while moving across different network architectures and to assess the impact on the communication between the MN and the content source introduced by the proposed framework. Although additional delay is introduced due to the *Resource Binding Procedure* (i.e., to discover a valid mapping for the content on the network architecture where the MN has moved to) and message conversion, in the evaluated scenario, after moving to a different network architecture the MN was able to continue downloading the content with a similar performance as in the scenario where the MN and the content were on the same network architecture. Specifically to the *Resource Binding Procedure*, the delay of such procedure is introduced only once after the handover procedure, but if a preemptive discovery before the handover is available, such delay to restore the communication could be even more reduced.

Despite the fact that the proposed solution was exemplified and evaluated using IP and NDN architectures, the designed solution is generic and flexible to accommodate other ICN instantiations and, eventually, other Future Internet network architectures.

4.5 Concluding Remarks

In this chapter, mobility frameworks in both evolutionary SDN-based and clean-slate ICN architectures, namely OpenFlow-based and ETArch architectures respectively, were presented with the purpose of optimizing link connectivity and flow establishment. For that, both frameworks were empowered with Media Independent Handover capabilities of the IEEE 802.21 standard allowing controlling entities to use context information from the link-layer of both MNs and network entities to efficiently manage and control the handover procedures of MNs. This allows handover-enhancement processes, provided by configurable indications from wireless link conditions and handover opportunities associated to MN movement, to dynamically and preemptively trigger the reconfiguration of the network regarding MN's flows. In doing so, it minimizes the impact to on-going data sessions and it increases connectivity opportunities. In both Future Internet architectural environments, results showed the benefits that both frameworks bring in terms of performance, when compared with more basic approaches. The MN's flows were able to be resumed immediately after the handover procedure, mitigating packet loss, but redundancy of packets during the handover procedure was verified since MN's flows were being duplicated towards the previous and the next point of attachments. Nevertheless, the presented frameworks are flexible to encompass different handover management strategies (either initiated, controlled and/or assisted by the MN or the network), allowing the optimization of different networking aspects.

Additionally, a novel inter-network architecture mobility framework was proposed which introduces a new entity, named Future Internet eXchange Anchor (FIXA), in the network. The FIXA acts as an anchor point and a gateway when the communication endpoints are on different network architectures, as well as it provides a procedure for MNs to discover how to access the content from a different network architecture after the handover. In doing so, MNs become capable of handover between access networks supporting different network architectures, without losing the reachability to content being accessed. Although this mobility framework was exemplified using NDN network architecture as the ICN instantiation, the proposed inter-network architecture mobility framework and mechanisms are generic and can be adapted to support other ICN instantiations.

Chapter 5

Conclusions

This final chapter summarizes the obtained results and provides key conclusions regarding the subjects researched in this thesis, pointing out potential directions for future work.

5.1 Conclusions

One of the key aspects responsible for the integration of Information and Communication Technologies (ICT) into an increasing number of societal areas worldwide as we see today, has been the flexibility and simplicity of the IP protocol. However, that same success has been continuously imposing new utilization requirements. Multimedia, Quality of Service (QoS), security and mobility are just examples of behaviors that are commonly granted today, but were not present at the conception of the IP protocol. As a result, research on the Future Internet has been gaining traction on recent years, with evolutions to the current Internet architecture as well as clean-slate architectures being proposed. Although differing on how architectures are designed, both approaches share the same goal: to provide better solutions to the current and expected future requirements of the Internet. With such a multitude of proposals, each focusing on specific key requirements, it can be expected that in the future an heterogeneity of protocols at the network layer might become a possibility, each targeting specific use case scenarios as already happens in the upper and lower layers. However, the deployment of a new network architecture may prove to be a costly and slowly process, with changes ranging from the network up to the application layer. This is more evident in what regards clean-slate architectures due to their disruptive nature.

While evolutionary approaches maintain compatibility with the legacy Internet architecture, allowing it to be incrementally deployed over the current architecture and to interoperate with the existing network environment, the deployment of clean-slate architectures may lead to information silos where information is restricted to each network architecture. Most of these clean-slate architecture instantiations do not take into consideration their compatibility in terms of interoperability with the existing IP networking environment, not defining mechanisms to allow entities to communicate outside each architecture. Instead, such architectures define that its transport can be made over the current IP network architecture to connect islands/hosts supporting a given network architecture, but still, the communication entities supporting different architectures is not envisioned. This communication between entities in different network architectures is a fundamental step for a fully functional heterogeneous Future Internet, where resources can be shared across different network architectures, avoiding the creation of information silos where information is restricted to a given network architecture.

Nevertheless, for a given network architecture to be widely adopted, it needs to properly address other emerging requirements, such as mobility. With the ever increasing number of mobile devices, with increasing capabilities and supporting different connectivity technologies, the traffic patterns on the Internet are changing. Much of the generated traffic in the Internet is not only addressed to mobile entities, but is also generated by those same mobile entities. User devices are becoming mobile instead of just being stationary, while acting not only as consumers of content but also as producers of content. This heterogeneous environment where different link technologies can be explored opens space for a new set of opportunities, by taking advantage of each link technology to provide an always best connected experience to the user. Context information from the current connection of the mobile node and from the neighboring environment as well as about the specificities of each data flow on the mobile node play an important role to achieve such goal. By having a more detailed knowledge about the underlying network, controlling entities (such, as mobility decision entities) can instruct mobile nodes to move to specific points of attachment, while preemptively reconfiguring the network to accommodate the new location of the MN prior to its handover.

In addition, the edge of the network may be a suitable location for the initial roll out of novel network architectures. Such heterogeneous networking environment at the edge of the network may introduce a new set of scenarios where a MN is able not only to move across access networks of the same network architecture but also to move across access networks of different network architectures. In both scenarios, it is expected that the MN could maintain its reachability to the resources being accessed.

The study on Future Internet architectures may help in identifying the key problems that need to be addressed by the existing Internet architecture as well as pointing out for possible solutions. Even if these architectures are not adopted as potential candidates for the future of the Internet, they may still provide valuable knowledge that could be leverage to improve the current IP network architecture. Moreover, the capability of addressing current and expected future Internet requirements is a key aspect for considering the adoption of a new network architecture. With interoperability and mobility gaining an increasingly importance in nowadays communications, it is important to study how the Future Internet, both evolutionary and clean-slate approaches, is able to address such requirements.

5.2 Review of Achievements

The main contributions of this thesis are three-fold: first, it contributes to the rolling out of novel Future Internet network architectures, by proposing an interoperability framework that enables entities in a given network architecture to transparently access resources deployed in a different one, allowing their incremental deployment in the existing networking ecosystem; second, it enhances Future Internet architectures, both SDN-based and ICN, with the capability of acquiring context information of the link-layer with the purpose of facilitating and optimizing the handover procedure of MN, by assisting the MN in the handover procedure as well as by preemptively reconfiguring the network to redirect MN ongoing flows towards the candidate point of attachments before the handover procedure itself; and third, it crosses both interoperability and mobility concepts enabling a new set of use case scenarios where a MN is able to not only move across access networks supporting the same network architecture but also move across access networks supporting different network architectures without losing the reachability to resources being accessed.

It is important to highlight that this thesis followed a more practical approach, with the proposed solutions being exemplified, adapted and implemented given the specificities of the selected network architectures. Such approach had the purpose of validating the correct operation of the different proposals, and assessing and evaluating their overhead and impact on the network operation. Nonetheless, the overall proposals in this thesis are generic and agnostic to any network architecture, being able of being adapted to support other network architectures.

An introduction to the base concepts regarding the research in the Future Internet was presented, briefly describing the operation of the most relevant Future Internet architectures, both SDN-based and ICN architectures, that were subject of study in this thesis. A study in the state of the art in what concerns the interoperability and mobility aspects was conducted to cover the two addressed research topics of this thesis.

In what concerns the research on interoperability research topic, it culminated with the proposal of an interoperability framework, named Future Internet Fusion (FIFu), that allows entities to access resources independently of the access network architecture of the communication endpoints. The FIFu framework follows a translation interoperability strategy, where messages are converted between architectures. An important aspect of this framework is that it is designed following a two-layer approach where the intelligent layer is responsible for controlling and managing the adaptation layer which in turn is responsible for the conversion of messages between architectures. Such layer separation enables a greater degree of flexibility to accommodate new protocols/architecture without requiring to step in on each individual equipment. This framework was evaluated through simulation and a proof-of-concept prototype, with results demonstrating the feasibility of the proposed framework. This evaluation considered scenarios where no interoperability was required for comparison purposes and, in some cases, the use of the proposed framework allowed to achieve better results than native IP approaches. As an added result, several key practical aspects were highlighted.

Although the proposed interoperability framework allowed a functional heterogeneous Future Internet environment, the adoption of novel network architectures is still dependent of being able to address other key requirements, such as mobility. In doing so, the work analyzed and contributed to mobility aspects in both evolutionary SDN-based and clean-slate ICN network architectures, resulting on the definition of IEEE 802.21-enabled mobility frameworks for OpenFlow-based architectures and for the Entity Title Architecture. These frameworks have in common the fact that they use link-layer context information provided by the IEEE 802.21 as input for the decisions about mobility procedures. This enabled both frameworks to preemptively reconfigure the flows of the MN towards the target PoA before the L2 handover procedure itself, allowing the MN to resume its ongoing sessions faster when compared with more naive approaches. The approach proposed in this thesis can potentially be adapted to other ICN architecture instantiations.

Finally, by integrating both interoperability and mobility concepts, novel mobility scenarios where MNs move across access networks supporting different network architectures becomes possible. To allow an holistic access to resources in such scenarios, an inter-network architecture mobility framework was designed. It introduces a new entity in the network, named Future Internet eXchange Anchor (FIXA), that acts as both anchor point and gateway whenever the MN moves to a different network architecture than the one where the content is deployed in. After the handover procedure of a given MN, the FIXA generates mappings to accessed content, that allows content to be addressed by the MN from its current network architecture. In the process, the FIXA acts as an anchor point whenever the communication endpoints are on different network architectures and as a gateway for converting messages between protocols supported on each network architecture. This framework was implemented and evaluated targeting IP and NDN network architectures in content retrieval use case scenarios. Such framework enables content being accessed by the MN to still be reached after moving to an access network of a different network architecture.

The achieved contributions regarding the work developed during the elaboration of this thesis had an impact in the key researched topics, which is proved not only by the accepted publications (both international conferences and journals) but also by the integration of the mobility concepts designed in this thesis in the EU funded FP7 OFELIA project. The most relevant publications are presented below:

Carlos Guimarães, José Quevedo, Rui Ferreira, Daniel Corujo, Rui L. Aguiar, *Exploring Interoperability Assessment for Future Internet Architectures Roll Out*, Journal of Network and Computer Applications, Vol. 136, pp. 38-56, June 2019

José Quevedo, Rui Ferreira, **Carlos Guimarães**, Rui L. Aguiar, Daniel Corujo, *Internet of Things discovery in interoperable Information Centric and IP networks*, Internet Technology Letters, Vol. 1, No. 1, 2018

Daniel Corujo, **Carlos Guimarães**, José Quevedo, Rui Ferreira, Rui L. Aguiar, *Information Centric Exchange Mechanisms for IoT Interoperable Deployment* book chapter in "User-Centric and Information-Centric Networking and Services Access Networks, Cloud and IoT Perspective" pp. 71-140, CRC Press, May 2019

Flávio Silva, Daniel Corujo, **Carlos Guimarães**, João Pereira, Pedro Rosa, Sergio Takeo, Augusto Neto, Rui L. Aguiar, *Enabling Network Mobility by Using IEEE 802.21 Integrated with the Entity Title Architecture*, Proc. 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribudos IV Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF), Brasilia, Brasil, May 2013

Carlos Guimarães, Daniel Corujo, Rui L. Aguiar, Flávio Silva, Pedro Rosa, *Empowering Software Defined Wireless Networks Through Media Independent Handover Management*, Proc. 2013 Globecom, Atlanta, USA, Dec 2013

Carlos Guimarães, Daniel Corujo, Flávio Silva, Pedro Rosa, Augusto Neto, Rui L. Aguiar, *IEEE 802.21-enabled Entity Title Architecture for Handover Optimization*, Proc. WCNC 2014 IEEE Wireless Communications and Networking Conference, Istanbul, Turkey, Apr 2014

Carlos Guimarães, Daniel Corujo, Rui L. Aguiar, *Enhancing OpenFlow with Media Independent Management Capabilities*, Proc. ICC 2014 IEEE International Conference on Communications, Sydney, Australia, Jun 2014

Carlos Guimarães, José Quevedo, Rui Ferreira, Daniel Corujo, Rui L. Aguiar, *Content Retrieval while Moving Across IP and NDN Network Architectures*, Proc. 24th IEEE Symposium on Computers and Communications (ISCC 2019), Barcelona, Spain, Jul 2019

5.3 Future Work

There is a number of possibilities for future research regarding the study of interoperability and mobility aspects in Future Internet architectures.

Future Internet architectures, and most specifically clean-slate architectures, still lack the definition and maturation of upper-layer protocols or intrinsic mechanisms to support different use case scenarios commonly taken for granted in the current Internet (e.g., web browsing and video streaming). Such support is critical for a widespread adoption of new network architectures. This had an impact on the design of the interoperability framework, where well-known protocols in the current IP network architecture were mapped to the base mechanisms provided by each network architecture (and vice versa). With the definition of specific-purposed protocols and applications of each network architectures to cope with a broader set of scenarios is expected that protocols on each architecture could better fit each other, which could improve the performance of the interoperability procedure.

The definition of mechanisms for specific use case scenarios may also improve how mappings between resource identifiers on each architecture are performed. By knowing the type of application, more suitable protocols can be chosen to be used in the foreign network architectures. The huge amount of mappings between architectures requires high-performance and scalable storage and resolution mechanisms to be put in place so that the performance of the interoperability procedure across architectures does not incur in additional delays.

The increasing capabilities of current mobile devices (e.g., smartphones) shift the role that user devices have in the communication, from being only consumers of content to be also producers of content (e.g., video calls). If we consider ICN architectures, MNs themselves can becomes caches of the consumed content, enabling them to provide the content to new requesters. As such, it is important to study how the proposed mobility mechanisms (e.g., preemptive configuration of flows) can impact on the mobility of mobile nodes when these act as producers of content.

Finally, by integrating IEEE 802.21 mechanisms with the proposed inter-network architecture mobility mechanisms, several optimizations become possible. If prior knowledge about the supported network architectures by each PoA is acquired before the handover procedure itself, the MN may leverage that information to handover to the most suitable handover candidate according to the application needs. Moreover, it also allows a preemptive discovery of content mappings to be used after the handover procedure, allowing applications to immediately reestablish the access to contents after the handover procedure.

Bibliography

[1] IEEE Standard for Local and metropolitan area networks - Media Independent Handover Services. *IEEE Std 802.21-2008* (January 2009), 1–323.

[2] IEEE Standard for Local and metropolitan area networks - Part 21: Media Independent Handover Services Amendment 2: Extension for Supporting Handovers with Downlink Only Technologies. *IEEE Std 802.21b-2012 (Amendment to IEEE Std 802.21-2008 as amended by 802.21a-2012)* (May 2012), 1–40.

[3] IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services - Amendment for Security Extensions to Media Independent Handover Services and Protocol. *IEEE Std 802.21a-2012 (Amendment to IEEE Std 802.21-2008)* (May 2012), 1–92.

[4] IEEE Standard for Local and metropolitan area networks– Part 21: Media Independent Handover Services - Amendment 3: Optimized Single Radio Handovers. *IEEE Std 802.21c-2014 (Amendment to IEEE Std IEEE Std 802.21-2008 as amended by IEEE Std 802.21a-2012 and IEEE Std 802.21b-2012)* (July 2014), 1–76.

[5] IEEE Standard for Local and metropolitan area networks – Part 21: Media Independent Handover Services Amendment 4: Multicast Group Management. *IEEE Std 802.21d-2015 (Amendment to IEEE Std 802.21-2008 as amended by IEEE Std 802.21a-2012, IEEE Std 802.21b-2012, and IEEE Std 802.21c-2014)* (July 2015), 1–110.

[6] aes, C. G., Corujo, D., de la Oliva, A., Ohba, Y., and Aguiar, R. L. Multicast group membership management in media independent handover services. *Computer Networks 62*, Supplement C (2014), 55 – 68.

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

[8] Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., and Willinger, W. Anatomy of a Large European IXP. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (New York, 2012), SIGCOMM '12, ACM, pp. 163–174.

[9] Aguiar, R. L. Some Comments on Hourglasses. *SIGCOMM Comput. Commun. Rev. 38*, 5 (September 2008), 69–72.

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

[11] Ali-Ahmad, H., Ouzzif, M., Bertin, P., and Lagrange, X. Distributed dynamic mobile IPv6: Design and evaluation. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)* (Shanghai, April 2013), pp. 2166–2171.

[12] Ali-Ahmad, H., Ouzzif, M., Bertin, P., and Lagrange, X. Distributed Mobility Management: Approaches and analysis. In *2013 IEEE International Conference on Communications Workshops (ICC)* (Budapest, June 2013), pp. 1297–1302.

[13] Atarashi, Y., Higuchi, H., and Tsuchiya, K. Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS). RFC 2767, February 2000.

[14] Azgin, A., Ravindran, R., and Wang, G. Mobility study for Named Data Networking in wireless access networks. In *2014 IEEE International Conference on Communications (ICC)* (Sydney, June 2014), pp. 3252–3257.

[15] Bao, C., Li, X., Baker, F., Anderson, T., and Gont, F. IP/ICMP Translation Algorithm. RFC 7915, June 2016.

[16] Barraca, J. P., Gomes, D., and Aguiar, R. L. AMazING - Advanced Mobile wIreless Network playGround. *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops* (2010).

[17] Bellier, L., Malki, K. E., Castelluccia, C., and Soliman, H. Hierarchical Mobile IPv6 (HMIPv6) Mobility Management. RFC 5380, October 2008.

[18] Bernardos, C. J., de la Oliva, A., Giust, F., and Ziga, J.-C. Proxy Mobile IPv6 extensions for Distributed Mobility Management. Internet-Draft draft-bernardos-dmm-pmipv6-dlif-00, Internet Engineering Task Force, October 2017.

[19] Berners-Lee, T., Fielding, R. T., and Masinter, L. M. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, January 2005.

[20] Bi, J., and Wang, Y. *Software Defined Mobility Management for Mobile Internet.* in Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture (eds M. Liyanage, A. Gurtov and M. Ylianttila), John Wiley & Sons, Ltd, 2015, pp. 265–287.

[21] Brim, S. W., and Carpenter, B. E. Middleboxes: Taxonomy and Issues. RFC 3234, February 2002.

[22] Carpenter, B. E., and Moore, K. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, February 2001.

[23] Chan, A., Liu, D., Seite, P., Yokota, H., and Korhonen, J. Requirements for Distributed Mobility Management. RFC 7333, August 2014.

[24] Chowdhury, K., Leung, K., Patil, B., Devarapalli, V., and Gundavelli, S. Proxy Mobile IPv6. RFC 5213, August 2008.

[25] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, September 2017.

[26] Colitti, L., Gunderson, S. H., Kline, E., and Refice, T. *Evaluating IPv6 Adoption in the Internet.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 141–150.

[27] Corujo, D., Figueiredo, S., and Aguiar, R. L. Media-independent multicast signalling for enhanced video performance in the MEDIEVAL project. In *2011 Future Network Mobile Summit* (Warsaw, June 2011), pp. 1–9.

[28] Corujo, D., Guimaraes, C., Santos, B., and Aguiar, R. L. Using an open-source IEEE 802.21 implementation for network-based localized mobility management. *Communications Magazine, IEEE 49*, 9 (2011), 114–123.

[29] Cui, Y., Sun, Q., Boucadair, M., ZOU), T. T. T., Lee, Y., and Farrer, I. Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture. RFC 7596, July 2015.

[30] Cui, Y., Wu, J., Wu, P., Vautrin, O., and Lee, Y. Public IPv4-over-IPv6 Access Network. RFC 7040, November 2013.

[31] Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., and Bailey, M. Measuring IPv6 Adoption. *SIGCOMM Comput. Commun. Rev. 44*, 4 (August 2014), 87–98.

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

[33] de Oliveira Silva, F., Gonalves, M. A., de Souza Pereira, J. H., Pasquini, R., Rosa, P. F., and Kofuji, S. T. On the analysis of multicast traffic over the Entity Title Architecture. In *2012 18th IEEE International Conference on Networks (ICON)* (Singapore, December 2012), pp. 30–35.

[34] Despres, R. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd). RFC 5569, January 2010.

[35] Durand, A., Droms, R., Lee, Y., and Woodyatt, J. Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion. RFC 6333, August 2011.

[36] Fasano, D. P., Guardini, D. I., Durand, A., and Lento, D. IPv6 Tunnel Broker. RFC 3053, January 2001.

[37] Feldmann, A. Internet Clean-slate Design: What and Why? *SIGCOMM Comput. Commun. Rev. 37*, 3 (July 2007), 59–64.

[38] Fernandez, M. P. Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)* (Barcelona, March 2013), pp. 1009–1016.

[39] Fisher, D. A Look Behind the Future Internet Architectures Efforts. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 45–49.

[40] Fonseca, R., Porter, G., Katz, R. H., Shenker, S., and Stoica, I. IP Options are not an option. In *Technical Report UCB/EECS- 2005-24, EECS Department, UC Berkeley* (December 2005).

[41] Fotiou, N., Islam, H., Lagutin, D., Hakala, T., and Polyzos, G. C. CoAP over ICN. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (Larnaca, November 2016), pp. 1–4.

[42] Fotiou, N., Nikander, P., Trossen, D., and Polyzos, G. C. Developing Information Networking Further: From PSIRP to PURSUIT. In *Broadband Communications, Networks, and Systems* (Berlin, Heidelberg, 2012), Springer Berlin Heidelberg, pp. 1–13.

[43] Fotiou, N., Trossen, D., and Polyzos, G. C. Illustrating a publish-subscribe Internet architecture. *Telecommunication Systems 51*, 4 (December 2012), 233–245.

[44] Fotiou, N., Xylomenos, G., Polyzos, G. C., Islam, H., Lagutin, D., Hakala, T., and Hakala, E. ICN Enabling CoAP Extensions for IP Based IoT Devices. In *Proceedings of the 4th ACM Conference on Information-Centric Networking* (New York, 2017), ICN '17, ACM, pp. 218–219.

[45] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., and Riviere, E. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev. 45*, 5 (September 2015), 37–42.

[46] Gavras, A., Karila, A., Fdida, S., May, M., and Potts, M. Future Internet Research and Experimentation: The FIRE Initiative. *SIGCOMM Comput. Commun. Rev. 37*, 3 (July 2007), 89–92.

[47] GENI. Global Environment for Network Innovations. http://www.geni.net/.

[48] Gilligan, R. E., and Nordmark, E. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213, October 2005.

[49] Giust, F., Bernardos, C. J., and de la Oliva, A. Analytic Evaluation and Experimental Validation of a Network-Based IPv6 Distributed Mobility Management Solution. *IEEE Transactions on Mobile Computing 13*, 11 (November 2014), 2484–2497.

[50] Giust, F., Cominardi, L., and Bernardos, C. J. Distributed mobility management for future 5G networks: overview and analysis of existing approaches. *IEEE Communications Magazine 53*, 1 (January 2015), 142–149.

[51] Giust, F., de la Oliva, A., and Bernardos, C. J. Flat access and mobility architecture: An IPv6 distributed client mobility management solution. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (Shanghai, April 2011), pp. 361–366.

[52] Giust, F., Oliva, A. D. L., Bernardos, C. J., and Costa, R. P. F. D. A network-based localized mobility solution for Distributed Mobility Management. In *2011 The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC)* (Brest, October 2011), pp. 1–5.

[53] Gleeson, T., Thaler, D., and Templin, F. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 5214, March 2008.

[54] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. NOX: Towards an Operating System for Networks. *SIGCOMM Comput. Commun. Rev. 38*, 3 (July 2008), 105–110.

[55] Gustafsson, E., and Jonsson, A. Always Best Connected. *IEEE Wireless Communications 10*, 1 (February 2003), 49–55.

[56] Handley, M. Why the Internet only just works. *BT Technology Journal 24*, 3 (July 2006), 119–129.

[57] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and Tokuda, H. Is It Still Possible to Extend TCP? In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (New York, 2011), IMC '11, ACM, pp. 181–194.

[58] Huang, B., Hui, D., and Savolainen, T. Dual-Stack Hosts Using "Bump-in-the-Host" (BIH). RFC 6535, February 2012.

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

[60] Johnson, D. B., Arkko, J., and Perkins, C. E. Mobility Support in IPv6. RFC 6275, July 2011.

[61] Jokela, P., Zahemszky, A., Esteve Rothenberg, C., Arianfar, S., and Nikander, P. LIPSIN: Line Speed Publish/Subscribe Inter-networking. *SIGCOMM Comput. Commun. Rev. 39*, 4 (August 2009), 195–206.

[62] Jung, C., and Carpenter, B. E. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529, March 1999.

[63] KIM, H., AND FEAMSTER, N. Improving network management with software defined networking. *Communications Magazine, IEEE 51*, 2 (2013), 114–119.

[64] KIM, S. M., CHOI, H. Y., PARK, P. W., MIN, S. G., AND HAN, Y. H. OpenFlow-based Proxy mobile IPv6 over software defined network (SDN). In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)* (Las Vegas, January 2014), pp. 119–125.

[65] KIM, Y.-J., SHIN, M.-K., LEE, S., DURAND, A., AND NORDMARK, E. Dual Stack Hosts Using "Bump-in-the-API" (BIA). RFC 3338, October 2002.

[66] KITAMURA, H. A SOCKS-based IPv6/IPv4 Gateway Mechanism. RFC 3089, April 2001.

[67] KOODLI, R. Mobile IPv6 Fast Handovers. RFC 5568, July 2009.

[68] KREUTZ, D., RAMOS, F. M. V., VERSSIMO, P. E., ROTHENBERG, C. E., AZODOLMOLKY, S., AND UHLIG, S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE 103*, 1 (January 2015), 14–76.

[69] KULINSKI, D., AND BURKE, J. NDN Video: Live and Prerecorded Streaming over NDN. NDN Technical Report NDN-0007, September 2012.

[70] LARA, A., KOLASANI, A., AND RAMAMURTHY, B. Network Innovation using OpenFlow: A Survey. *IEEE Communications Surveys Tutorials 16*, 1 (First 2014), 493–512.

[71] LEINER, B. M., CERF, V. G., CLARK, D. D., KAHN, R. E., KLEINROCK, L., LYNCH, D. C., POSTEL, J., ROBERTS, L. G., AND WOLFF, S. A Brief History of the Internet. *SIGCOMM Comput. Commun. Rev. 39*, 5 (October 2009), 22–31.

[72] LIANG, J., JIANG, J., DUAN, H., LI, K., WAN, T., AND WU, J. When HTTPS Meets CDN: A Case of Authentication in Delegated Service. In *2014 IEEE Symposium on Security and Privacy* (San Jose, May 2014), pp. 67–82.

[73] LIU, Z., WU, Y., YUEPENG, E., GE, J., AND LI, T. Experimental evaluation of consumer mobility on named data networking. In *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)* (Shanghai, July 2014), pp. 472–176.

[74] MASTORAKIS, S., AFANASYEV, A., MOISEENKO, I., AND ZHANG, L. ndnSIM 2: An updated NDN simulator for NS-3. Technical Report NDN-0028, Revision 2, NDN, November 2016.

[75] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (March 2008), 69–74.

[76] Melazzi, N. B., Detti, A., Arumaithurai, M., and Ramakrishnan, K. K. Internames: A name-to-name principle for the future Internet. In *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine), 2014 10th International Conference on* (Rhodes, August 2014), pp. 146–151.

[77] Moiseenko, I., and Oran, D. TCP/ICN: Carrying TCP over Content Centric and Named Data Networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (New York, 2016), ACM-ICN '16, ACM, pp. 112–121.

[78] Moiseenko, I., Stapp, M., and Oran, D. Communication Patterns for Web Interaction in Named Data Networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking* (New York, 2014), ACM-ICN '14, ACM, pp. 87–96.

[79] Muscariello, L., Carofiglio, G., Auge, J., and Papalini, M. Hybrid Information-Centric Networking. Internet-Draft draft-muscariello-intarea-hicn-00, Internet Engineering Task Force, June 2018.

[80] Nguyen, T. T., Bonnet, C., and Harri, J. SDN-based Distributed Mobility Management for 5G Networks. In *2016 IEEE Wireless Communications and Networking Conference* (Doha, April 2016), pp. 1–7.

[81] Nguyen, X. N., Saucez, D., Barakat, C., and Turletti, T. Rules Placement Problem in OpenFlow Networks: A Survey. *IEEE Communications Surveys Tutorials 18*, 2 (Second quarter 2016), 1273–1286.

[82] Niebert, N., Baucke, S., El-Khayat, I., Johnsson, M., Ohlman, B., Abramowicz, H., Wuenstel, K., Woesner, H., Quittek, J., and Correia, L. M. The way 4WARD to the creation of a future internet. In *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications* (Cannes, September 2008), pp. 1–5.

[83] Oliva, A. D. L., Banchs, A., Soto, I., Melia, T., and Vidal, A. An overview of IEEE 802.21: media-independent handover services. *IEEE Wireless Communications 15*, 4 (August 2008), 96–103.

[84] Pan, J., Paul, S., and Jain, R. A survey of the research on future internet architectures. *Communications Magazine, IEEE 49*, 7 (2011), 26–36.

[85] Perkins, C. E. IP Mobility Support for IPv4, Revised. RFC 5944, November 2010.

[86] Postel, J. NCP/TCP transition plan. RFC 801, November 1981.

[87] Pupatwibul, P., Banjar, A., Sabbagh, A. A., and Braun, R. Developing an Application Based on OpenFlow to Enhance Mobile IP Networks. In *38th Annual IEEE Conference on Local Computer Networks - Workshops* (Sydney, October 2013), pp. 936–940.

[88] Rahman, A., Trossen, D., Kutscher, D., and Ravindran, R. Deployment Considerations for Information-Centric Networking (ICN). Internet-Draft draft-irtf-icnrg-deployment-guidelines-03, Internet Engineering Task Force, June 2018.

[89] Ravindran, R., Chakraborti, A., Amin, S. O., Azgin, A., and Wang, G. 5G-ICN: Delivering ICN Services over 5G Using Network Slicing. *IEEE Communications Magazine 55*, 5 (May 2017), 101–107.

[90] Ravindran, R., Suthar, P., Trossen, D., and White, G. Enabling ICN in 3GPP's 5G NextGen Core Architecture. Internet-Draft draft-ravi-icnrg-5gc-icn-02, Internet Engineering Task Force, July 2018.

[91] Raza, S. M., Kim, D. S., Shin, D., and Choo, H. Leveraging Proxy Mobile IPv6 with SDN. *Journal of Communications and Networks 18*, 3 (June 2016), 460–475.

[92] Ren, J., Lu, K., Wang, S., Wang, X., Xu, S., Li, L., and Liu, S. VICN: a versatile deployment framework for information-centric networks. *IEEE Network 28*, 3 (May 2014), 26–34.

[93] Rexford, J., and Dovrolis, C. Future Internet Architecture: Clean-slate Versus Evolutionary Research. *Commun. ACM 53*, 9 (September 2010), 36–40.

[94] Stuckmann, P., and Zimmermann, R. European research on future Internet design. *IEEE Wireless Communications 16*, 5 (October 2009), 14–22.

[95] Tantayakul, K., Dhaou, R., and Paillassa, B. Impact of SDN on Mobility Management. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)* (Crans-Montana, March 2016), pp. 260–265.

[96] Thomas, Y., Tsilopoulos, C., Xylomenos, G., and Polyzos, G. C. Multisource and Multipath File Transfers Through Publish-subscribe Internetworking. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking* (New York, 2013), ICN '13, ACM, pp. 43–44.

[97] Thomas, Y., Tsilopoulos, C., Xylomenos, G., and Polyzos, G. C. Accelerating File Downloads in Publish Subscribe Internetworking with Multisource and Multipath Transfers. In *WTC 2014; World Telecommunications Congress 2014* (Berlin, June 2014), pp. 1–6.

[98] Trossen, D., and Parisis, G. Designing and realizing an information-centric internet. *IEEE Communications Magazine 50*, 7 (July 2012), 60–67.

[99] Trossen, D., Reed, M. J., Riihijrvi, J., Georgiades, M., Fotiou, N., and Xylomenos, G. IP over ICN - The better IP? In *Networks and Communications (EuCNC), 2015 European Conference on* (Paris, June 2015), pp. 413–417.

[100] Tyson, G., Sastry, N., Cuevas, R., Rimac, I., and Mauthe, A. A Survey of Mobility in Information-centric Networks. *Commun. ACM 56*, 12 (December 2013), 90–98.

[101] Tyson, G., Sastry, N., Rimac, I., Cuevas, R., and Mauthe, A. A Survey of Mobility in Information-centric Networks: Challenges and Research Directions. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications* (New York, 2012), NoM '12, ACM, pp. 1–6.

[102] Valtulina, L., Karimzadeh, M., Karagiannis, G., Heijenk, G., and Pras, A. Performance Evaluation of a SDN/OpenFlow-Based Distributed Mobility Management (DMM) Approach in Virtualized LTE Systems. In *2014 IEEE Globecom Workshops (GC Wkshps)* (Austin, December 2014), pp. 18–23.

[103] Waddington, D. G., and Chang, F. Realizing the transition to IPv6. *IEEE Communications Magazine 40*, 6 (June 2002), 138–147.

[104] Wang, L., Moiseenko, I., and Zhang, L. NDNlive and NDNtube: Live and Prerecorded Video Streaming over NDN. NDN Technical Report NDN-0031, April 2015.

[105] Wang, S., Bi, J., Wu, J., Yang, X., and Fan, L. On Adapting HTTP Protocol to Content Centric Networking. In *Proceedings of the 7th International*

Conference on Future Internet Technologies (New York, 2012), CFI '12, ACM, pp. 1–6.

[106] Wang, Y. PMIPv6-based Partially Distributed Mobility Management Modeling and Evaluation. In *2016 World Automation Congress (WAC)* (Rio Grande, July 2016), pp. 1–7.

[107] Wang, Y., and Bi, J. A Solution for IP Mobility Support in Software Defined Networks. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)* (Shanghai, August 2014), pp. 1–8.

[108] Wu, P., Cui, Y., Wu, J., Liu, J., and Metz, C. Transition from IPv4 to IPv6: A State-of-the-Art Survey. *IEEE Communications Surveys Tutorials 15*, 3 (Third 2013), 1407–1424.

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

[110] Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., and Polyzos, G. C. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys Tutorials 16*, 2 (Second 2014), 1024–1049.

[111] Yamamoto, K., and ichiro Hagino, J. An IPv6-to-IPv4 Transport Relay Translator. RFC 3142, June 2001.

[112] Zhang, G., Li, Y., and Lin, T. Caching in information centric networking: A survey. *Computer Networks 57*, 16 (2013), 3128 – 3141. Information Centric Networking.

[113] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. Named Data Networking. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 66–73.

[114] Zhang, Y., Afanasyev, A., Burke, J., and Zhang, L. A survey of mobility support in Named Data Networking. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (San Francisco, April 2016), pp. 83–88.

[115] Zuniga, J. C., Bernardos, C. J., de la Oliva, A., Melia, T., Costa, R., and Reznik, A. Distributed mobility management: A standards landscape. *IEEE Communications Magazine 51*, 3 (March 2013), 80–87.