BONVOYAGE

From Bilbao to Oslo, intermodal mobility solutions, interfaces and applications for people and goods, supported by an innovative communication network

Research and Innovation Action GA 635867

Deliverable D3.1:

Networking

Deliverable Type:	R/Other
Deliverable Number:	D3.1
Contractual Date of Delivery to the EU:	31.10.2016
Actual Date of Delivery to the EU:	29.10.2016
Title of Deliverable:	Networking
Work package contributing to the Deliverable:	WP3
Dissemination Level:	PU

Editor:	Giuseppe Piro [CNIT]
Author(s):	Nicola Blefari Melazzi [CNIT], Piero Boccadoro [CNIT], Andrea Detti [CNIT], Luigi Alfredo Grieco [CNIT], Giuseppe Piro [CNIT], Giuseppe Ribezzo [CNIT], Giuseppe Tropea [CNIT]
Internal Reviewer(s):	Stephan Strodl and Roman Pickl [FLUIDTIME], Dag Kjenstad [SINTEF]
Abstract:	This deliverable describes the BONVOYAGE communication system. Specifically, it focuses on the following main aspects: (i) reference scenario and targeted network architecture, (ii) Information-centric Networking and Internames, and (iii) networking functionalities integrated in the BONVOYAGE platform. Finally, a proof-of-concept implementation showing the delivering of travel-centric contents through the request-response communication scheme is presented too.
Keyword List:	BONVOYAGE communication system, networking, Information-Centric networking, Internames, Middleware, Namespace

Table of Contents

List of Figures

List of Tables

Abbreviations

BONVOYAGE Abbreviations	
Abbreviation	**Definition**
API	Application Programming Interface
App	Application
CCN	Content-Centric Networking
COMET	COntent Mediator architecture for content-aware nETworks
DONA	Data Oriented Network Architecture
FIA	Future Internet Assembly
GPS	Global Positioning System
GTFS	General Transit Feed Specification
HTTP	HyperText Transfer Protocol
ICN	Information-Centric Networking
IoT	Internet of Things
IP	Internet Protocol
IRN	Internames Rendezvous Node
ISL	Internames Service Layer
ITS	Intelligent Transportation System
NDN	Named Data Networking
NetInf	Network of Information
NPRA	Norwegian Public Roads Administration
NRS	Name Resolution Service
OGB	OpenGeoBase
ORS	Object Resolution Service
P2P	Peer-to-Peer
PURSUIT	Publish-Subscribe Internet Technology
SAIL	Scalable & Adaptive Internet solutions
RH	Resolution Handler
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

Table 1: Abbreviations

BONVOYAGE Glossary

Table 2 lists and describes the terms that have been considered relevant in this deliverable.

BONVOYAGE Glossary	
Term	**Definition**
Bonvoyage Communication System	Communication bus available in the BonVoyage platform. It integrates a heterogeneous network architecture, the Internames Service Layer offering standardized and technological-independent networking functionalities, and logical nodes implementing search engine, routing, and rendezvous functionalities. It allows to retrieve contents through request-response and publish-subscribe primitives.
Data consumer	End user of the BonVoyage platform interested to the information for planning and optimizing transportation services.
Data producer	Entity of the BonVoyage platform generating and publishing travel-centric information and/or participatory sensing data.
Information-Centric Networking (ICN)	Emerging paradigm for the Future Internet, where all information (e.g., a picture) are identified by names that do not include references to its location.
Internames	Customized instance of ICN, which identifies by names all elements involved in network communication, including contents, services, users, and devices. Therefore, it enables name-to-name communications in heterogeneous network.
Internames Rendezvous Node (IRN)	Logical node of the BonVoyage Communication System performing publish-subscribe functionalities.
Internames Service Layer (ISL)	The middleware integrated in the BonVoyage Communication System, implementing Internames functionalities.
Name Resolution Server (NRS)	Logical node of the BonVoyage Communication System performing routing functionalities.
Named Data Networking (NDN)	is an ICN oriented project. It is characterized by a hierarchical naming scheme, coupled request and data paths and by-name routing for requests and secure packets with signature.
Object Resolution Server (ORS)	Logical node of the BonVoyage Communication System performing search engine functionalities.

Table 2: BONVOYAGE Dictionary

1 Introduction

1.1 Deliverable rationale

This deliverable describes the **BONVOYAGE Communication System**, developed by WP3 in the context of the *Task 3.1 - Networking*.

The BONVOYAGE Communication System is in charge of disseminating travel-centric data and participatory sensing contents over a heterogeneous network architecture embracing different communication protocols. The developed solution leverages the Information-Centric Networking (ICN) paradigm and offers the possibility to retrieve data by using request-response and publish-subscribe mechanisms, while ensuring the communication requirements identified for the targeted transportation system.

This document investigates networking-related aspects, including protocol interoperability, inter-domain routing, naming, caching, and low-level communication primitives. Also it explains the standardized Application Programming Interfaces (APIs) driving the communication process and describes the preliminary version of the testbed showing the main functionalities of the proposed solution.

All the technical details reported in this document will drive the the activities related to *Task 3.2 - Publish/Subscribe and security functionality* and *Task 3.3 - Travel-centric and participatory sensing services*.

1.2 Quality review

The internal reviewers responsible of this deliverable are Stephan Strodl and Roman Pickl (FLUIDTIME) and Dag Kjenstad (SINTEF).

1.3 Executive summary

1.3.1 Deliverable description

Starting from the BONVOYAGE architecture presented in the project proposal, and also depicted in Figure 1, this deliverable investigates the reference scenario that the BONVOYAGE Communication System is called to support. Specifically, Section 2 firstly provides general definitions of the data producer (i.e., who provides travel-centric contents or sensing data), the data consumer (i.e., who is interested in that information), and the underlying communication technology. Then, by taking into account the distributed and mobile nature of data sources, the coexistence of heterogeneous network protocols, and the requirements characterizing a journey planning platform, it also explicitly demonstrates the need for an innovative information

Version Control Table			
Version n.	**Purpose/Changes**	**Author**	**Date**
0.1	First draft	CNIT	10 October 2016
0.2	WP3 internal review	CNIT	21 October 2016
0.99	Peer Review	Stephan Strodl and Roman Pickl [FLUIDTIME], Dag Kjenstad [SINTEF]	28 October 2016
1.0	Final check. Authorized	Nicola Blefari Melazzi	31 October 2016

Table 3: Version Control Table

delivery paradigm, which goes beyond the current Internet architecture.

The resulting BONVOYAGE Communication System leverages and properly extends emerging communication technologies, currently identified within the umbrella of the so-called Information-Centric Networking (ICN) paradigm, which are being developed for building the so-called Future Internet. The description of its main facets starts from Section 3, which provides an overview on ICN principles and provides technical details on the concrete ICN architectures proposed so far. Particular attention is devoted to Internames, a customized ICN instance enabling name-to-name communications in a heterogeneous network architecture. Among all the available solutions, in fact, Internames emerged as a very promising scheme that natively satisfies the communication requirements envisaged in the BONVOYAGE project. Thus, Internames has been properly extended with Pub/Sub functionalities in order to fulfill the requirements of the targeted intelligent transportation system. Architectural details, design choices, and networking functionalities have been presented in Section 4. The conceived approach integrates the following aspects:

1. at the network layer, it is assumed to have a heterogeneous architecture, where nodes could be grouped in clusters, simply referred to as network realms, where the data delivery is handled by means of specific communication protocols; these realms are connected to each other by means of border routers implementing advanced networking functionalities;

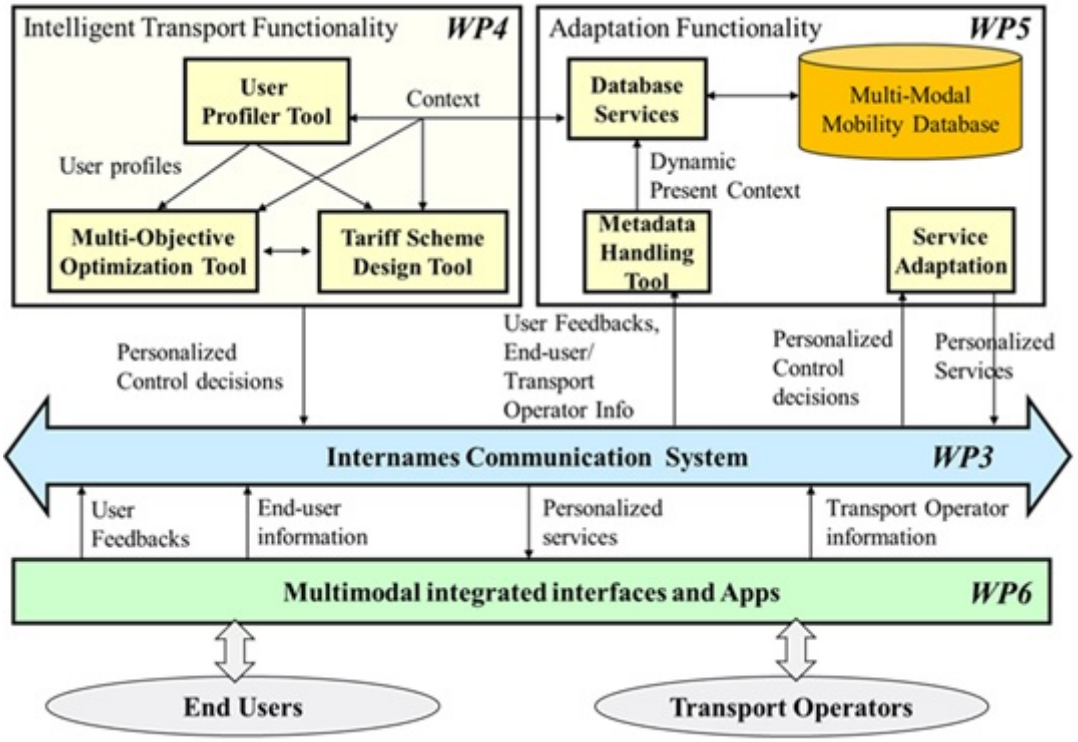

Figure 1: The preliminary architecture of the BONVOYAGE platform

2. Internames is deployed on top of the aforementioned network architecture for orchestrating name-to-name communications;

3. all the elements involved in network communication, including contents, services, users, and devices are identified by a name; the name-space has been designed to uniquely describe the characteristics of contents and services available within the aforementioned heterogeneous network architecture;

4. to better support a wide range of applications with different requirements, the content exchange is handled through both request-response and publish-subscribe communication models. Therefore, the BONVOYAGE communication system exposes request-response and publish-subscribe primitives to upper layers;

5. logical nodes are integrated in the Internames architecture for implementing search engine, routing, and rendezvous services;

6. all networking operations (and in particular request-response and publish-subscribe primitives) are exposed to the application layer through a standardized and scalable interface (e.g., the middleware layer, also referred to as Internames Service Layer). The Internames Service Layer is in charge of hiding all the details of the underlying communication tech-

nology and simplifying the interoperability among coexistent and heterogeneous network domains.

The request-response communication mechanism is deeply investigated in Section 5 with concrete example (or a concrete example) and further demonstrated in Section 6 through a preliminary experimental implementation.

1.3.2 Summary of results

In summary, this deliverable provides the following results:

- the general definition of the reference scenario, communicating peers, and the underlying communication technology;
- the discussion of communication challenges that motivated the design of an innovative information delivery paradigm, which goes beyond the current Internet architecture;
- the review of the state of the art of concrete network architectures based on the ICN principles (including the Internames instance);
- the high-level description of the BONVOYAGE Communication System;
- the provisioning of some preliminary considerations on routing and caching algorithms to be integrated in the BONVOYAGE platform;
- the deeply investigation of request-response and publish-subscribe primitives in the BONVOYAGE Communication System;
- a presentation of the preliminary implementation of the experimental testbed.

2 Reference scenario and target network architecture

2.1 Introduction to *open* journey planning platforms

The BONVOYAGE project aims at designing, developing, and testing a platform for optimizing multimodal and door-to-door transport of passengers and goods. The term multimodal is used to indicate that the platform jointly integrates travel information coming from different transport means and operators, thus offering unified planning and ticketing services. Moreover, the keyword door-to-door is used to remark that the goal is to find a (multimodal) travel route able to connect any couple of geographical points (without leaving uncovered, for example, the path from the passengers' home to the nearest train station). Based on these definitions, *how does BONVOYAGE differ from current solutions?*

Journey planning websites and services generally play a central role in helping transport operators and public authorities to increase the quality, the ridership, and the revenues of transportation services. These instruments, however, have been traditionally conceived as closed systems and the provisioning of journey planning services has been mostly delegated to a small number of specialized providers.

More recently, the tendency to publish and share transportation data directly on the web is emerging. For instance, in this context Google is trying to define a standardized format, namely General Transit Feed Specification (GTFS), for exposing public transportation schedules and related geographic information [1]. Also, another possible solution is DATEX format, 2nd version, namely DATEX II [2]. This new trend is laying its basis for an emerging open model for journey planning which brings two fundamental innovations. While various suppliers, thus including one-man app developers, could be able to implement their own applications for journey planning, at the same time the access to a wide range of travel-centric data will natively support inter-modality and interoperability. This happens because the journey configuration is simplified and promoted encompassing different means of transportation, solutions, and domains.

In addition, as we live in the Internet of Things (IoT) era, the pervasive integration of sensors into cars, trucks, and other transportation units (either public or private), which can be networked and used for traffic and road conditions monitoring, predicting arrival and departure information, ensuring traffic safety, and so on is evident. As a result, it is expected that a huge number of open and crowd-sourced data will shortly be adopted in a generic Intelligent Transport System (ITS). Coming from websites, data-centers, sensors, vehicles, goods and people on the move, all of these information can significantly improve the overall journey planning services.

BonVoyage Communication System capitalizes on these emerging trends: it targets the

design and the implementation of an open journey planning platform where external players, transport operators, sensors, passengers and other data sources may publish and use travel data to plan journeys. Of course, this leads to the creation of a unified journey planning platform, supporting the interaction among such a large number of actors, across the boundaries of organizations and domains, resulting in a very powerful enabler of a continent-wide smart-transportation system.

The continental door-to-door traveling system targeted by the BonVoyage Communication System project entails an almost constant exchange of information between travelers, tools, databases, multi-modal transportation systems, sensor networks, and forecasting models. The travel optimization, in fact, is performed by simultaneously considering several inputs like:

- non-real time characteristics (e.g., coverage, routes, schedules, type of goods, etc.) of the candidate transportation means (e.g., public transport such as bus, train, boat, taxi, airplane; private transport such as car, bicycle, walking, cooperative modes like car-sharing, trucks);
- real-time requirements (e.g., availability of private transport means for horizontal, user-to-user sharing, traffic congestion, temporary road barriers, lane closures, temporary speed limits, new stops, available space and weight to complete the load etc.);
- user profiles (including user preferences, needs and expectations);
- users' feedback (also in real time and under the form of participatory sensing);
- dynamic tariff schemes.

The reference scenario is depicted in Figure 2 and shows that the BonVoyage Communication System integrates many heterogeneous actors providing both static (e.g., a GTFS time-table produced by a transport operator) and real-time (e.g., vehicle speed captured from a sensing device) data, which can be used by any intelligent journey planner application. The proposed scenario includes different transport operators, many sensors that provide real-time and crowd-sourced information supporting the travel-centric services, a fully distributed "P2P" bike-sharing journey planner App, and the main BONVOYAGE App that fully exploits all of the platform's capabilities, side-by-side with a traditional GPS navigation application, and consisting of a backend Engine and a frontend App. For instance, transport and freight operators are connected to BonVoyage Communication System and ask for information regarding timetables, availability, transportation, practicability, delays or any other travel-related information. Information providers offer context information to the platform in order to enrich its intelligent

transport capabilities. Distributed sensors can provide information about weather conditions, quality of the air, traffic congestions and so on. Also, as booking still remains one of the main need felt by travelers, while availability is confirmed, ticketing services and navigation systems are furnishing data useful to complete these operations.

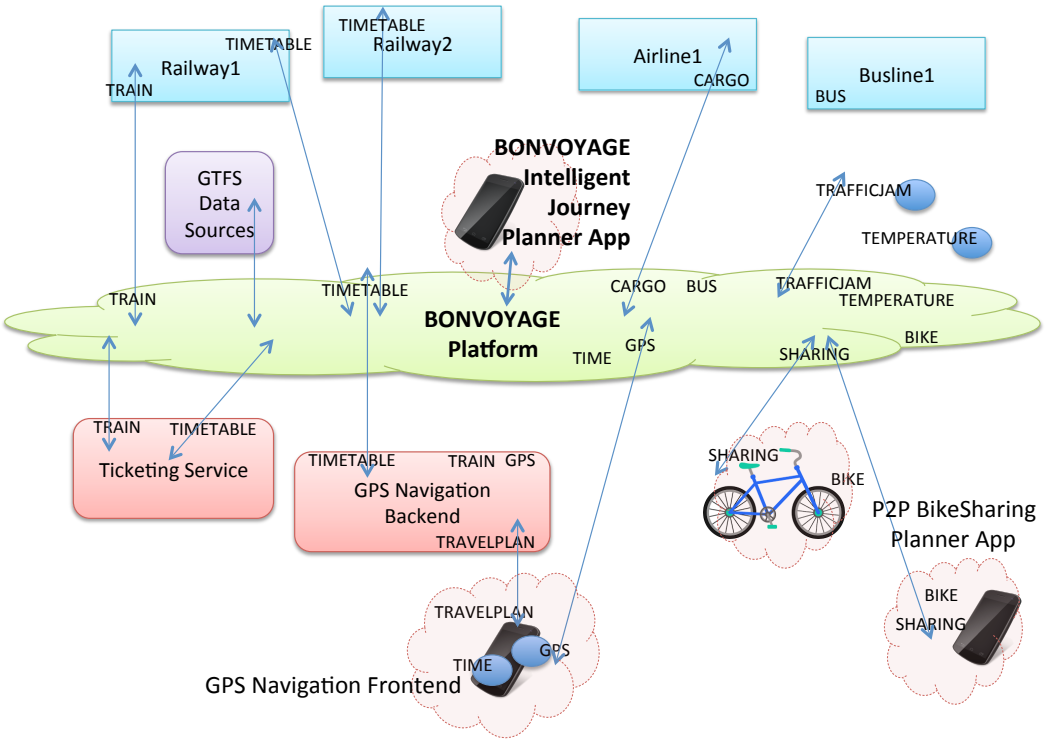

Figure 2: Interactions among heterogeneous actors in the BONVOYAGE platform.

Entities forming the entire ecosystem previously depicted are divided into two main categories:

- **Data Producer**, i.e., the actor that generates and publishes travel-centric and sensing data in the platform, and
- **Data Consumer**, i.e., the end users of the BonVoyage Communication System that are interested to the aforementioned information for planning and optimizing transportation services.

For instance, transport operators, traffic lights, webcams, and sensors cover the role of data producers. Journey planners and travelers, instead, act as data consumers. As data production and data usage are operation continuously performed in a worldwide environment, it might also happen that an entity comes to cover, at the same time, the role of both data producer and data

consumer. Indeed, passengers traveling on public means of transport are both data consumer (at first) and data producers (as long as they use mobile app or other shared platform to provide feedback). Private drivers are passengers of private means of transportation: they initially cover the role of data consumer. As long as BonVoyage Communication System is also monitoring traffic, quality of service and other similar variables, feedback from private drivers can be part of the working scenario and can provide useful information to all users. As a consequence, private drivers also act as data providers in the considered system.

Note that the introduction of these categories is extremely important for better understanding the communication process and the list of atomic operations handled at the network layer and presented in the rest of this document.

2.2 Big picture of the underlying communication network

Given the high heterogeneity of data producers and data consumers, it is expected that the BonVoyage Communication System should be deployed on top of a **heterogeneous network architecture**, where many nodes are grouped in clusters (namely **network realms**). In each cluster, the data delivery is performed by means of specific communication protocols. An example is provided in Figure 3, where a journey planner (i.e., the data consumer) and transportation agencies of Rome (e.g., a set of data producers) are attached to different network realms. More specifically, at the right side of the figure there is an entity that publishes travel-centric data (i.e., a set of time-tables related to buses available in Rome). At the left side, there is the end user interested to that information. Without loss of generality, the figure also depicts an overlay communication system, namely the BonVoyage Communication System, that orchestrates data delivery across multiple domains.

What immediately emerges from this preliminary description of the underlying communication network is that the design of an efficient communication system for journey planning is urgent. At the time of this writing, in fact, no concrete solutions could be exploited for building up an effective communication bus for the BonVoyage Communication System. This is not so just for transport operators but also researchers, industries, and public authorities involved in the ITS domain now or in the near future.

In this context, many requirements have to be fulfilled.

It is mandatory to work toward a platform that allows efficient **content-routing** as network has to be able to notify people when new information are available. Moreover, using unique identifiers, users have to be able to perform a GET operation to reach the information they are searching for.

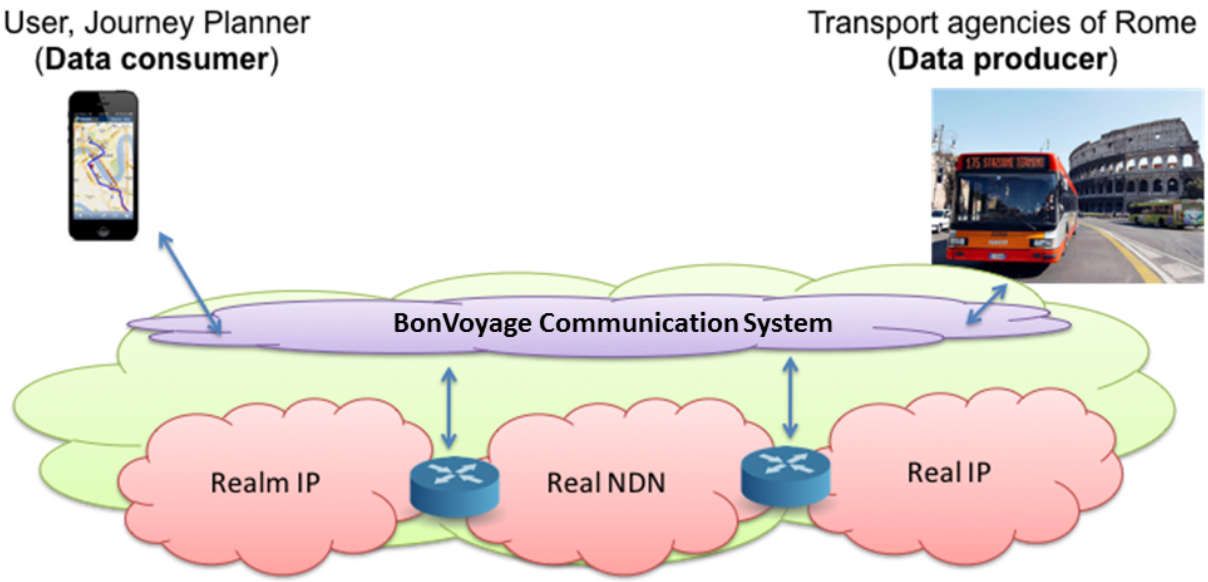

Figure 3: Baseline network architecture

The communication system is also required to be **transparent** and **scalable**. The first requirement is for operation taking place within the network, meaning that retrieving data is happening at the application layer through a standardized interface, which hides all the details of the technological solutions implemented.

Scalability is not only referred to as for the number of contents, thus involving data storage, but also for Network realms dimensions and number of devices involved. In this way, the core components can be implemented over groups of logical nodes, on which we have to focus our attention.

To achieve the overall specifications, the new architecture has to **improve efficiency**; this can be obtained through caching operations. As users, data producers and data consumers are all on the move but connected to the network, accessibility becomes the third parameter. It is really easy to understand how much this is gaining importance, as caching can be used in strategically identified places or roles. Moreover, the concept of mobility suggests that as users change point of attachment to the network, the task performed is still a simple GET request as the interesting part of the information is the next chunk of the content. This has to be

communicated without maintaining active paths or re-routing from an anchor point. If needed, the chunk has to be provided by a different node than the one used before, thus depicting a content-aware network.

As communications described are **not synchronized**, but decoupled in time and space, the system has to be able to provide content at these conditions. This will help communication from/to sensors networks, ad-hoc networks, vehicular networks, social gatherings, mobile networks on board trains, planes, bikes. To better understand the operating scenario, as an hopefully not frequent situation, it is important to imagine networks stricken by disaster/interruptions.

Data consumer and Data producer are not connecting directly one to the other so their communication is seen as **peer-to-peer**, thus hiding the overall system and making the information simply available.

Somehow, these communications have to be **secure**, not only according to the communication itself, as it is easy to understand that data and information retrieved have to be consistent and safe. A number of issues are afflicting the current Internet, not only consisting of the hackers and crackers community working but concerns remains when thinking about data safety and authenticity. The ideal solution to this is data becoming self-consistent and self-certifying, also guaranteeing users' privacy.

Not only access but even updates and removal of contents, when obsolete or expired, have to be allowed, so it becomes clear that access control has to be content-oriented. When data are related to or coming from different services they need to be differentiated and this is going to be useful only when furnished without complex or high-layer procedures.

Contents have to be created and disseminated in a modular way for better customization and **granular distribution**.

2.3 Do we really need an innovative information delivery paradigm?

The distributed and mobile nature of data sources, as well as the coexistence of heterogeneous network protocols, requires an innovative information delivery paradigm, which goes beyond the current Internet architecture. Accordingly, the BONVOYAGE project promises to design an innovative communication system based on the **information-centric paradigm**. Before providing more technical details on it, it could be useful to understand the reasons that are at the basis of the need for an innovative information delivery paradigm.

Internet is a global communication infrastructure that hosts billions of devices and enables the exchange of Exabyte of data per year [3]. According to its native design, the communication

between clients (i.e., the data consumer) and remote servers (i.e., the data producer) entails the establishment of end-to-end connections. This data exchange mechanism can be considered as **host-centric** in nature: any information is strictly related to the location of nodes that make them available, as generally expressed by the Uniform Resource Locator (URL) or the Internet Protocol (IP) address. Such a conventional paradigm was considered a winning strategy since the creation of packet-switched networks, but the current network usage is extraordinarily increasingly requiring the adoption of different communication paradigms. Nowadays, in fact, users are more interested in sharing contents rather than in interacting with remote devices. As a result, they are experiencing a novel **content-centric** usage of the Internet. It is important to note that this trend in also fully integrated in the BonVoyage Communication System: data producers would publish travel-centric and sensing data; data consumers would retrieve these data without needing to interact with remote servers.

At the time of this writing, there exist many solutions that could be used for better supporting content distribution and service provisioning, while trying to overcoming the limitation of the classic host-to-host Internet model. They include (just to name a few) Content Delivery Networks, Peer-to-Peer architectures, and cloud-based platforms [4] [5]. Unfortunately, conventional communication schemes generate, in this context, heavy issues in terms of scalable content distribution, mobility, security and trust, resilience of ICT services to system failures, and content availability [4]. Also the Future Internet Assembly (FIA) has identified some fundamental limitations of the current Internet, classifying them in four main areas [6] [7]: (i) the impossibility to process and handle, in real-time, data packets exchanged within the network according to the information stored within them; (ii) the lack of content/context aware caching and storage capabilities in network routers; (iii) the inefficient transmission of content-oriented data; and (iv) the lack of flexibility and adaptive control mechanisms that could react in an autonomous manner to external events.

On the contrary, emerging services and applications, including the BONVOYAGE platform, would leverage a communication system able to ensure the following aspects [8]:

1. **availability**: an aspect connected to users needing to fetch contents in a fast, reliable, and effective way;

2. **location-independence**: there is no need for many (neither for all) services to identify the location of a given data;

3. **security**: in continuity with the previous focusing point, it is necessary to trust contents independently from the location and the identity of who is providing them;

4. **mobility**: it is necessary to guarantee the seamless support of mobile users which should not experience any service interruption or discontinuity when moving across different access points belonging to different networks;

5. **scalability**: the problems related to limited storage, bandwidth, connectivity and computational capabilities that affect service providers when handling a huge number of users should not influence the behavior, the system and the resulting quality of services;

6. **fault-tolerance**: for all the application fields it is mandatory to increase the resilience of ICT services to system failures.

In order to satisfy the aforelisted requirements, the **Information-Centric Networking (ICN)** has emerged as a revolutionary formulation: it intends to re-think the foundation of the current Internet by introducing flexible and efficient mechanisms for content dissemination [9]. In its main rationale, ICN envisages that: (i) each content is addressed through a name that does not contain anymore a reference to its location; (ii) the user interested in a specific content just sends a request including the name of the desired data item; (iii) the request is processed by the network based on specific name resolution and/or routing-by-name approaches.

Given the reference scenario, **name-based communication** proves itself to enable advanced ITS applications where both sensor and human-generated information can be utilized for a smart-transportation environment. Allowing name-based communication becomes mandatory to create a working environment in which door-to-door transportation comes true, also becoming able to guarantee Quality of Service and location-independent access. For this reason, the BONVOYAGE project aims at designing an innovative communication system based on ICN principles.

3 Information-Centric Networking and Internames

The need of ICN has been mainly motivated by the inefficiency of traditional host-centric networks in delivering contents, and the foreseen difficulty they will have in meeting future user expectations, with increasing demand for online video (or multimedia) exchange and large files transfer.

ICN focuses on the content (i.e., what is relevant to the user) rather than on who is providing the content (and thus where the content is located), and targets **receiver-driven communications**, based on **content names**, **name-based routing**, and **self-certifying packets**. Hence, a user, namely a data consumer, expresses an interest for a content. Then, the ICN architecture takes care of routing-by-name the content request towards the best source (that could be the data producer, a replica server or an in-network cache) and sends the related data back to the user [10].

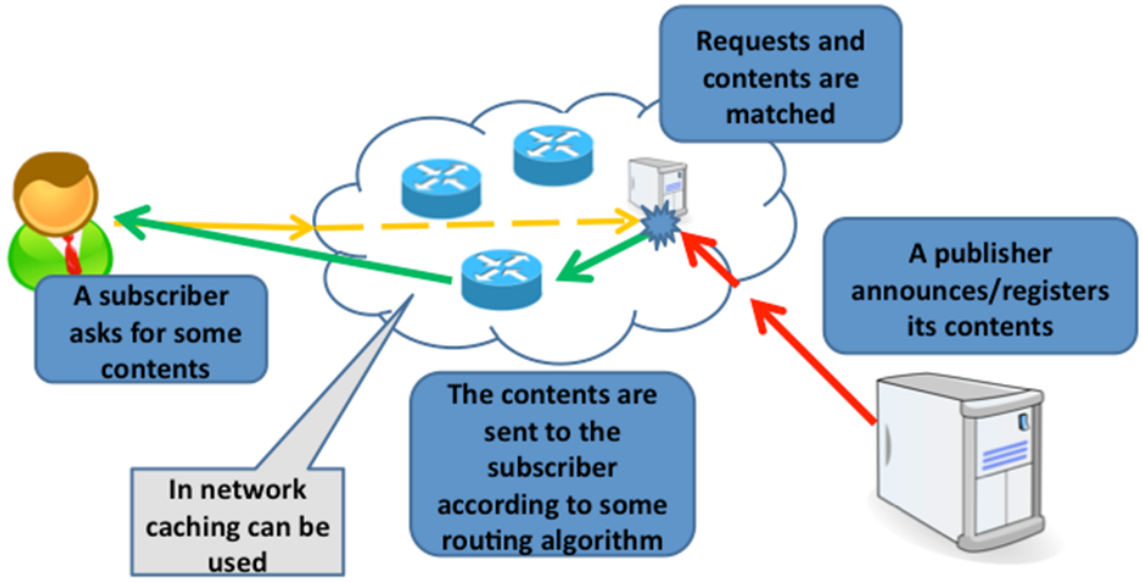

Figure 4: Generic ICN model

Figure 4 describes a generic ICN model: the publisher announces his/her contents; a data consumer asks for some content (i.e., he/she issues a subscription); requests and announcements are matched; routing primitives set up the path to deliver data items from the closer subset of nodes, which are in possession of the information asked for. All these operations are driven by names, which means that locators can be still used but only when this is really necessary and (in any case) no global addressing scheme for hosts is required anymore.

In summary, ICN offers the following advantages:

1. **Efficient content-routing**: ICN would enable Internet Service Providers (ISPs) to perform native content routing. This would be a built-in facility of the network, which would transform the Internet to a native content distribution network, as opposed to Content Delivery Networks (CDN) overlays, which are the current practice.

2. **In-network caching**: caching enabled today by HTTP proxies requires performing complex stateful operations and are increasingly difficult owing to the fact that more and more traffic is nowadays encrypted at the source. ICN would significantly improve efficiency and scalability of caching by judiciously using caches at critical points in the network, exploiting the knowledge of the name of content.

3. **Simplified handling of mobile and multicast communication**: with ICN, unlike current mobility architectures, when a user changes the point of attachment to the network, she will simply ask for the next chunk of the content she is interested in, without the need for maintaining tunnels or re-routing from an anchor point and maintaining state in the network; the next chunk may be provided by a different node than the one that would have been used before the handover. Furthermore, multicast is an inherent capability in ICN, with content requested from a node being delivered to all interested receivers, without using overlays. This applies to mobile networked devices, such as sensors and actuators, as well.

4. **Simplified support for time/space-decoupled communications**: allowing fragmented networks, or sets of devices to operate even when disconnected from the rest of the network (e.g. sensors networks, ad-hoc networks, vehicular networks, social gatherings, mobile networks on board trains, planes, or networks stricken by disaster/interruptions).

5. **Simplified support for peer-to-peer communications**: ICN can inherently support communications between peers, without the need for application-layer overlays, as is the practice in current-day networks.

6. **Content-oriented security model**: securing the content itself, instead of securing the communication channels, allows for a more flexible and customizable protection of content and user privacy and protects in-network caches from fake content.

7. **Content-oriented access control**: ICN can provide access to content as a function of time, place (e.g., country), or profile of user requesting the item. Likewise, this functionality allows implementing: i) access revocation (also known as digital forgetting), so that

content may be removed from the system by its creator, ii) garbage collection, deleting from the network expired/obsolete contents.

8. **Content-oriented quality of service differentiation (and possibly pricing)**: ICN would enable ISPs to differentiate the quality perceived by users from different services without complex, high-layer procedures.

9. **Creation and dissemination of contents in a modular and personalized way**: ICN provides opportunities for better customization of the interests of users and the content that is published by providers. This will enable more efficient consumption of content because of better granularity in how content is described and identified.

3.1 Big picture of concrete ICN architectures

At the time of this writing, ICN oriented projects include Data-Oriented Network Architecture (DONA) [11], Publish Subscribe Internet Technology (PURSUIT) [12], Scalable & Adaptive Internet solutions (SAIL) [13], COntent Mediator architecture for content-aware nETworks (COMET) [6], CONVERGENCE [14], and Named-Data Networking (NDN) [15]. Even if these architectures share the ICN communication paradigm (presented before), they differently address some other key aspects, including: naming schema and security, routing strategies, cache management (see the Appendix for more detail). A concise outline of the listed ICN architectures and the related features is provided in Table 5. In a dedicated section of the present document, Internames will be described in details.

As regards to naming schema and security aspects, two main approaches have been explored. The NDN project adopts hierarchical human-readable naming scheme. Names are composed by multiple strings of characters, arranged in a URL fashion. In general, a trusted relationship between such names and the matching data is required. To this aim, NDN appends to the returned data packet a signature (generated from a concatenation of name and information included in the message itself) and details about the key used to produce the signature (for instance, the public key of the signer, and a certificate for that public key). Instead, DONA, PURSUIT, SAIL, NetInf, and MobilityFirst projects rely on a flat self-certifying structure of names: each data is identified through the hash of its name, authenticated by using the public key of the data producer. Differently from NDN, flat naming schemes natively integrate security aspects. The CONVERGENCE architecture supports both flat and hierarchical naming schemes. Finally, the COMET project introduces URI-based names, which are generally self-certified.

	ICN Architectures							
	DONA	NetInf	PURSUIT	SAIL	CONVERGENCE	NDN	Mobility First	COMET
Naming Scheme	Flat	Flat	Flat	Flat	Flat and Hierarchical	Hierarchical	Flat	URI-based
Routing of requests	Resolution handler nodes (in hierarchical structure)	Resolution Exchange Entity	Rendevouz nodes (in hierarchical Structure)	Name Resolution System (in hierarchical Structure)	Name Resolution System (in hierarchical Structure)	Routing-by-name	Global Name Resolution Service	Content Resolution System (in hierarchical Structure)
Data Path	Route is created during data resolution towards a specific address within network	Packets are forwarded towards a specific address	Data path is created by topology manager	Route is created during data resolution towards a specific address within network	Packets are sent back through reverse request path	Packets are sent back through reverse request path	Ruled by Global Name Resolution (no name resolution process dependency)	Route created during resolution
Routing of Requests vs data path	Coupled OR Decoupled	Decoupled	Decoupled	Coupled OR Decoupled	Coupled	Coupled	Decoupled	Coupled
Security	Self-signed names AND packets authentication	Self-signed names	Self-signed names AND packets authentication	Self-signed names	Signature within objects	Signature within packets	Self-signed names	Self-signed names
Caching	On-path scheme (Resolution handler) OR Off-path scheme	Off-path scheme through (additional) registration	Off-path scheme through (additional) registration	On-path scheme by network routers OR Off-path scheme enabled	On-path scheme by network routers OR Off-path scheme enabled	On-path scheme by network routers OR Off-path scheme enabled	On-path scheme by network routers OR Off-path scheme enabled	Probabilistic On-path scheme by Network Routers OR Off-path scheme

Figure 5: Summary of ICN Architectures

ICN solutions also differ in term of routing mechanisms, used for both name resolution and data path creation. Name resolution usually refers to the mapping of a content name to possible locators of content sources. The data path, instead, identifies the set of intermediate routers through which the data packet is forwarded towards the user that generated the request. Note that these two operations may be coupled (i.e., data are sent back to the user through the same path of the request) or decoupled (i.e., requests and corresponding data packets follow different routing paths, often created by different logical nodes). Coupled approaches are adopted in CONVERGE, NDN, and COMET. Decoupled schemes are integrated in NetInf, PURSUIT, and Mobility first proposals. DONA and SAIL support both of them.

ICN architectures diverge also in caching mechanisms used for storing content copies within network nodes. On-path and off-path caching strategies have been investigated so far. The on-path caching supposes that a requested data can be stored and later retrieved by any intermediate node visited by the content request. This scheme is natively supported by ICN architecture where name resolution and data path are offered in a coupled fashion, e.g., NDN and CONVERGENCE. With the off-path strategy, replicas of a requested content can be found in network nodes not crossed by the user request. In both cases, the ICN architecture uses nodes with cached content as parallel data producers, which may be taken into account during the name resolution process and/or routing-by-name operations.

In the rest of this Deliverable, only PURSUIT and NDN architectures will be used to describe networking primitives of the BONVOYAGE Communication System. For the final

implementation, instead, only the NDN architecture will be taken into account.

3.2 Internames

In the big picture described, Internames plays a central role, as it makes full integration and support of ICN solutions possible. Here, communication becomes name-to-name centric, rather than host-to-name based, and to that aim, an abstraction layer that enables communications among mobile users is needed and therefore Internames is not designed on the specific network architecture and/or structure. For the sake of completeness, the following paragraphs will clarify what is Internames made of and how it works so integration aspects will be analyzed underlining the reasons why it is designed to use names not only within an ICN network realm but over any network architecture.

First of all, it is of great importance to specify that Internames is able to separate identifiers (names) and locators (addresses). Names are dynamically mapped with a specific attention to time, location, context and service. Internames also exposes to applications a name-oriented API (see Figure 6) that enables to retrieve content identified by name. Moreover, it is possible pushing data towards the communication interface (port) of an application (i.e., a service access point) that is identified by a name.

In the Figure 7 the main entities are depicted. First of all we have the Name-realms. The namespace is formed by name realms that are (in the more general case) disjoint containers of names, managed by different administrations. A generic name (e.g., n2n://nriA:Alice.com/cell) is a URI composed by a name-realm identifier (e.g. nriA) followed by an identifier that uses the local naming scheme (e.g. Alice.com/cell), or, in some cases, a set of names or even the set of current DNS names or flat identifiers. Network-realms are also reported. A named-entity is dynamically (or statically) bound to one or more Network Attachment Points (NAPs) (possibly available in different network-realms). A network-realm is an autonomous network using a local networking stack (e.g., IP, Ethernet, ICN, etc.), and whose routing scope is bounded to that network domain only. It is also possible that a network-realm is nested, in which case it uses the networking services of the underlying realm to interconnect its nodes (which implies the formation of an overlay network).

Object Resolution Service (ORS) are entities configured as searchable database containing all the names of the namespace together with related metadata. Users searching for content would query the ORS (as with current search engines) and obtain a name or list of names.

Name Resolution Service (NRS) and Routing Resolution Service (RRS) are used to manage routing operations. NRS handles inter-realm routing. RRS, instead, manages intra-realm

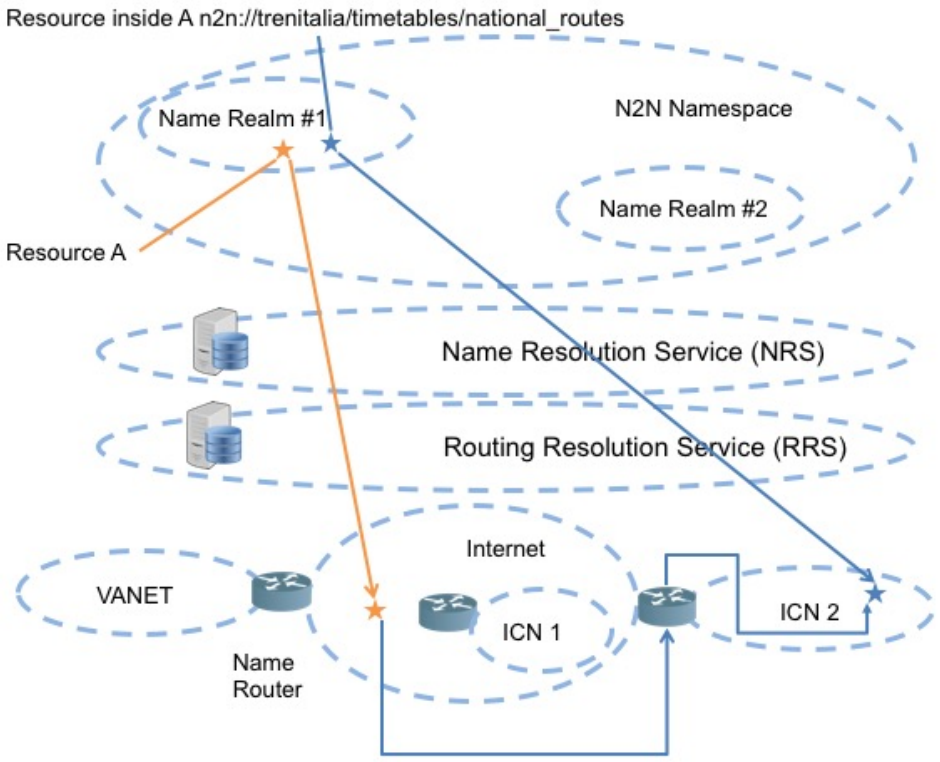

Figure 6: Internames architecture components

routing.

Network-realms are interconnected to each other through dedicated nodes called name-routers (NR). To support the flexible integration of the architecture with current protocols, a name-router can act as a protocol proxy between the name-oriented protocols operating in each of the interconnected network-realms.

In Internames, the forward and the reverse path do not need to be the same. Since a bidirectional flow could traverse different network realms, the flow in the forward and backward direction have to be resolved through the source/destination name. In order to be backward compatible with NDN and/or utilize the additional feature provided by routers in an NDN realm where the network provider chooses to enable PIT, Internames can use the same path for the forward and reverse direction with the correct combination of gateway to the NDN realm and service descriptor.

In the Figure 7 all the aforementioned components are working together in a structured communication sequence. The interaction of a user with the infrastructure starts with a query to the ORS. Users send a list of keywords which finds one or more objects. Each of them is

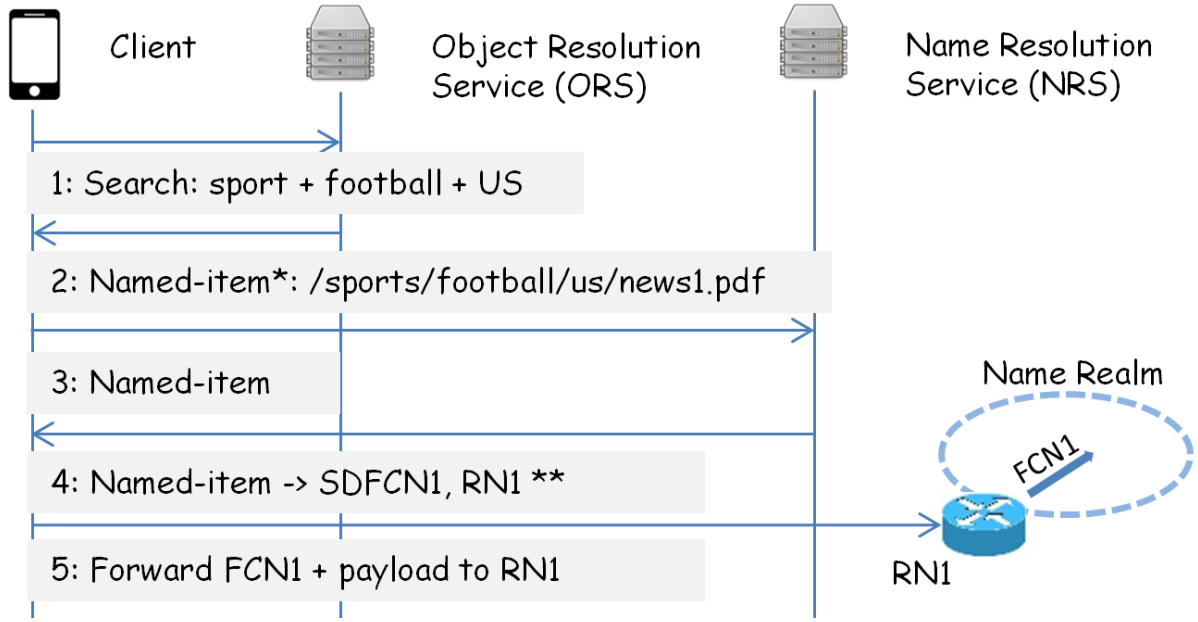

Figure 7: Message flow in Internames

associated with a named entity, then returned to the user. The user chooses one of them and asks the Name Resolution System (NRS) to resolve the current location of the named-entity in terms of its network-realm (realm id), its locator within the realm (dst-loc) and the protocol means to reach it within the realm (service descriptor: SD). The locator of the named-entity complies with the network technology used in that network-realm. To transfer a message towards the destination realm (through one or more transit network-realms) the RRS provides the locator of the next hop name-router towards the destination realm and the related name-oriented protocol to be used to reach it.

4 The BONVOYAGE Communication System

Even if ICN is posing the foundation of the Future Internet, it is widely accepted that it is impossible to reach a worldwide network architecture completely based on ICN in upcoming years. Only few portions of the network could be really re-designed from scratch, thus integrating pure ICN communication principles (clean-state solution [16]). Nevertheless, to push the evolution of the Internet towards the information-centric approach, ICN could be quickly deployed as an overlay network, i.e., on top of the existing IP protocol suite [17].

Without loss of generality, many other pieces of the current Internet architecture will still use the conventional IP protocol. Thus, the resulting network architecture taken into account in the BONVOYAGE project is extremely **heterogeneous**. From one side, network nodes will be grouped in clusters, simply referred to as **realms**, where data delivering is handled by means of specific communication protocols. From another side, network realms will be connected to each other through dedicated border routers, thus requiring advanced internet-working solutions.

Starting from these premises, the BONVOYAGE Communication System has been conceived in order to target:

- the **design** of a scalable communication protocol based on the ICN principles, able to easily support the exchange of travel-centric and sensing contents across multiple domains;
- the simple **provisioning of request-response and publish-subscribe communication schemes** on top of the aforementioned heterogeneous network, while ensuring a complete interoperability among different technologies and applications; Request-response and publish-subscribe communication schemes are the main mechanisms through which applications may fetch contents from the network [18]. With the request-response scheme, data are retrieved synchronously: every time a journey planner application would fetch that content, it has to send a request towards an external source of information, which will immediately provide the corresponding answer. The publish-subscribe communication model, instead, is based on an asynchronous interaction. In that case, the journey planner application issues a subscription request for a given content of interest. Then, every time a new content is generated, the external source of information is in charge of delivering that data to all the subscribed applications.
- the **development** of a scalable name-space through which uniquely identifying travel-centric information, sensing data, data consumers, data producers, and any operation handled at both application and network layers.

In order to support name-based communications across heterogeneous domains and data exchange in this multi-domain environment, the communication system has been designed by properly extending a specific instance of ICN, namely "**Internames**" [19]. Specifically, in Internames, all elements involved in the network communication (including contents, services, users, and devices) are identified by names. Moreover, dedicated logical nodes are disseminated in the network in order to offer search engines, message routing across realms, and publish-subscribe functionalities.

Then, to support data dissemination in a way that is transparent from the underlying communication technology and that guarantees a standardized interaction among different applications, a powerful middleware layer, namely **Internames Service Layer**, is deployed on top of the network layer. In particular, any device involved in the communication, like data consumers, data producers, border routers, and logical nodes implements an instance of the Internames Service Layer that (i) exposes a set of standardized APIs for requesting, subscribing, and publishing contents to the application layer and (ii) hides all the operations executed at the network layer when triggered by aforementioned APIs.

Finally, a hierarchical name-space has been designed to identify contents, users, border routers, and Internames logical nodes, as well as any operations triggered at the network layer by the aforementioned Internames APIs.

Note that the entire BONVOYAGE platform jointly exploits Internames, the middleware layer and the name-space for efficiently offering request-response and publish-subscribe communication mechanisms on top of a heterogeneous network architecture.

A big picture of the BONVOYAGE platform and its communication system is depicted in Figure 8. Instead, more technical details will be provided in the following sub-sections.

4.1 Logical nodes of Internames

Internames evolves from the baseline ICN **host-to-name** principle to a **name-to-name** principle, in which names can identify not only contents, but entities involved in the communication and services, without requiring a statically bound to a physical location. Moreover, source/destination entities and services are included [19]. Nevertheless, even if it natively allows the data exchange across heterogeneous network realms (e.g., this represents one of the goal of the BONVOYAGE project), only the request-response communication scheme is envisaged in its preliminary formulation. Therefore, it has been extended also for taking into account publish-subscribe mechanisms.

In short, Internames offers search engines, message routing across realms, and publish-

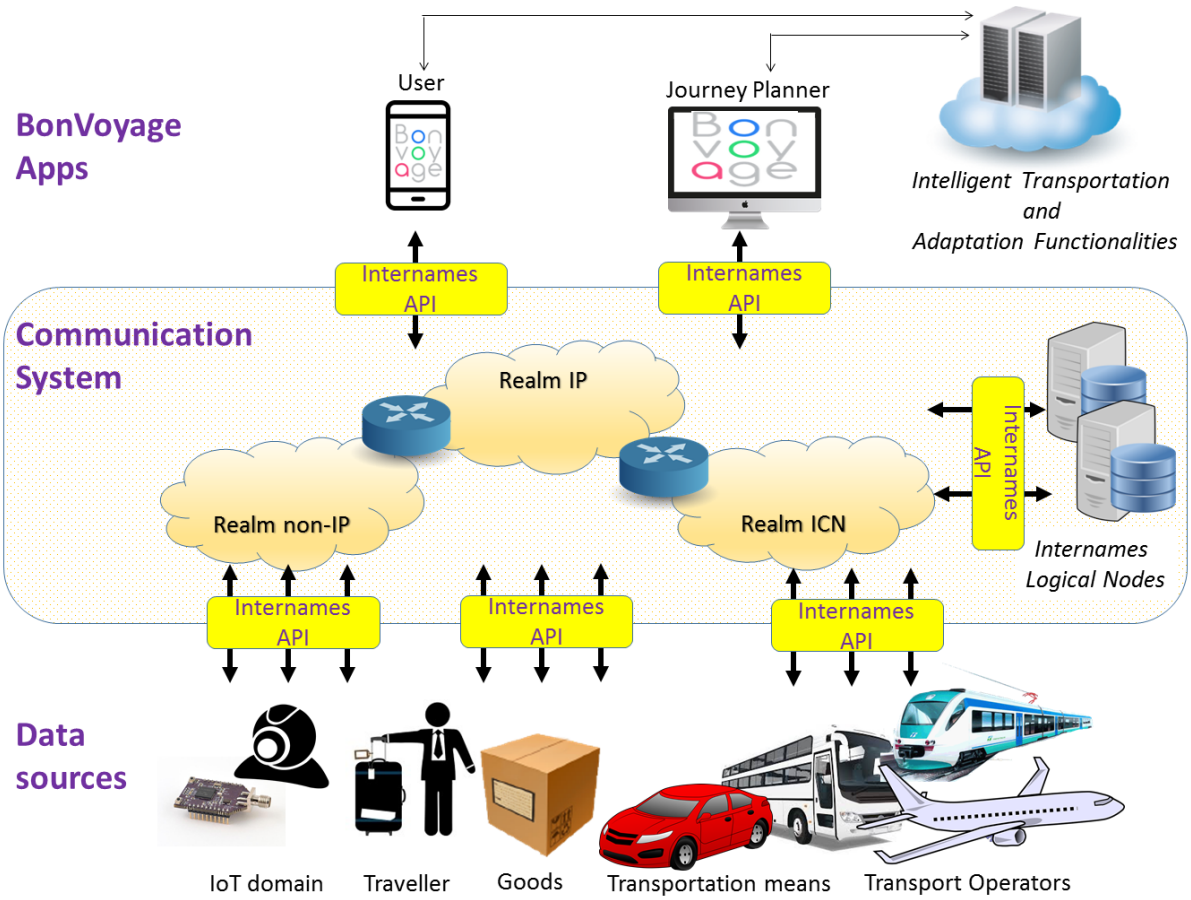

Figure 8: The BONVOYAGE Communication System

subscribe functionalities through three logical nodes: **ORS**, **NRS**, and **the Internames Rendezvous Node (IRN)**. ORS, NRS, and IRN are used to execute three different high-level functionalities offered at the network layer. A sketch of these functionalities is reported in Figure 9.

ORS offers a mapping functionality and plays the rule of a search engine. In general, a data consumer that is interested to retrieve a given content could not know a priori the name that the BONVOYAGE platform uses to uniquely identify it. Therefore, it sends to the ORS node a message containing a set of meta-data. The ORS will answer with a message reporting the list of names that map the aforementioned meta-data. Since ORS implements functionalities that are equal or very similar to those already provided by Google (or any other search engine) in today's Internet, no further implementation details will be investigated in this paper.

NRS handles routing operations within the heterogeneous network infrastructure. By knowing the distribution of data providers, it finds the best routing path through which forwarding the requests generated by data consumers. NRS is contacted by the data consumer or by other routers along the path every time a request should be sent towards a new network realm. This

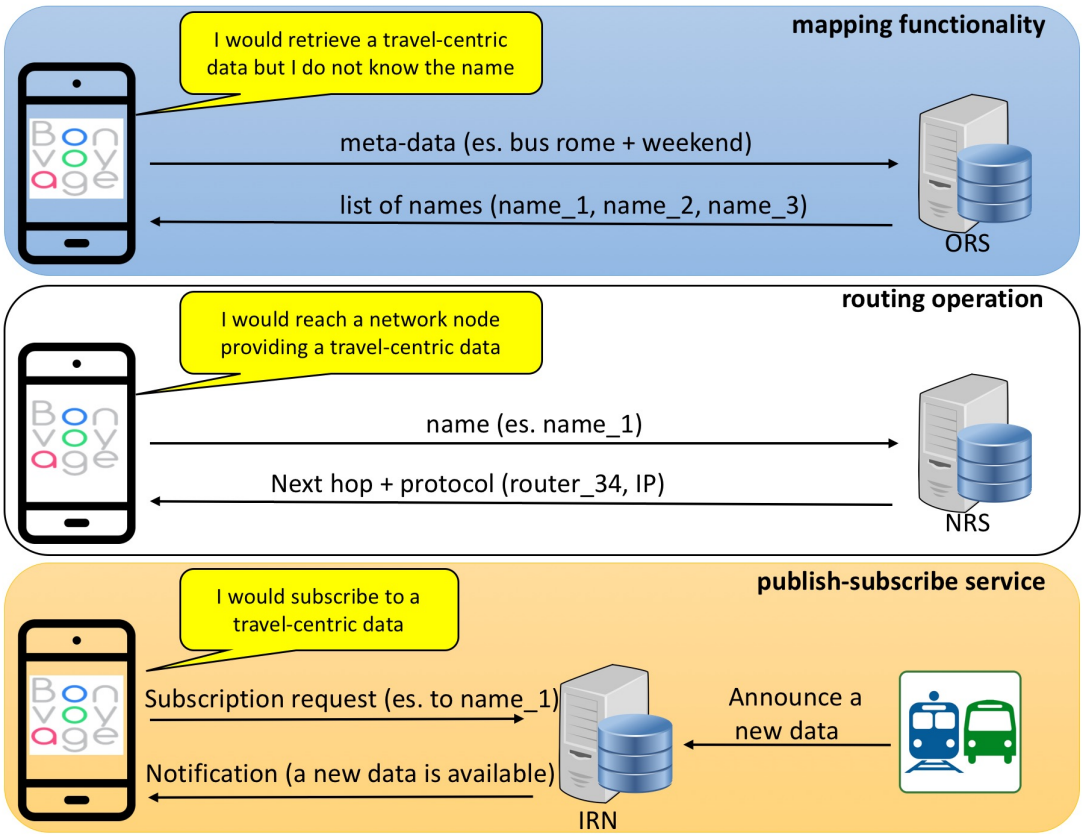

Figure 9: High-level functionalities offered at the network layer

routing functionality receives as input the name of the content to be retrieved and returns as output the next hop of the path and the related network protocol to use.

Finally, IRN manages publish-subscribe functionalities. Hence, it tracks the set of contents available in the BONVOYAGE platform, processes the submission requests, and notifies subscribed data consumers every time a new content of interest is available.

The IRN architecture and implementation detail are going to be presented in Deliverable, D3.2 (Publish/Subscribe and security functionality).

4.2 The Internames Service layer

In BONVOYAGE, the high-level functionalities illustrated before are implemented as technology-agnostic solutions. In fact, given the heterogeneity of the BONVOYAGE communication infrastructure, as well as the distributed deployment of its logical nodes, the provisioning of the high-level functionalities described before is a complex task to accomplish. To this end, the **Internames Service Layer** has been designed to offer all of these functionalities to the application layer by means of a standardized interface.

To better understand all of the concepts used from now until the end of this document, it is important to focus our attention on two entities, massively used and recalled here: **middleware** and **API**. Middleware is the name given to a software solution providing services to software applications enriching those available from the operating system. It is designed to glue together different solutions and helps software developers to perform communication and input/output, so it is easier to develop the specific application. Connecting software components together it lies between the operating system and the applications on each side of a distributed computer network. As a design solution, it includes Web servers, application servers, content management systems, and similar tools supporting application development and delivery. According to the fact that it is commonly used to enable communication and management of data in distributed applications, the BONVOYAGE platform is clearly the most suitable case for development and integration. An API, instead, is definable as a set of routines, protocols, and tools for building software and applications, where a software component is defined in terms of its operations, inputs, outputs, and underlying types and functionalities. These have to be considered as independent from their respective implementations, as it is for technological background on which the various kind of networks are based. API implementation and development allows definitions and implementations without compromising the interface neither implying a new design process. As the development of a good API makes it easier to design a program providing all the building blocks, using an API to make it possible for different solution to work together is part of the operative requirements. Due to the fact that an API can be used either for a web-based system, an operating system or database system, how useful it becomes obvious. This is enforced by the possibility for the API to be used to develop GUI components, interesting as they are the end-user side for both data producers and data consumer when working on the mobile device on which they are using the application.

Based on these premises, the Internames Service Layer acts as a middleware between upper layer applications and underlay communication technologies. As described in Figure 10, that middleware is distributed among all the actors of the BONVOYAGE platform (i.e., data consumers, data producers, network routers, and logical nodes of BONVOYAGE Communication System). Thus, it aims at hiding the technical details related to low-level networking operations and making the implementation of upper level applications independent from the underlying communication technology.

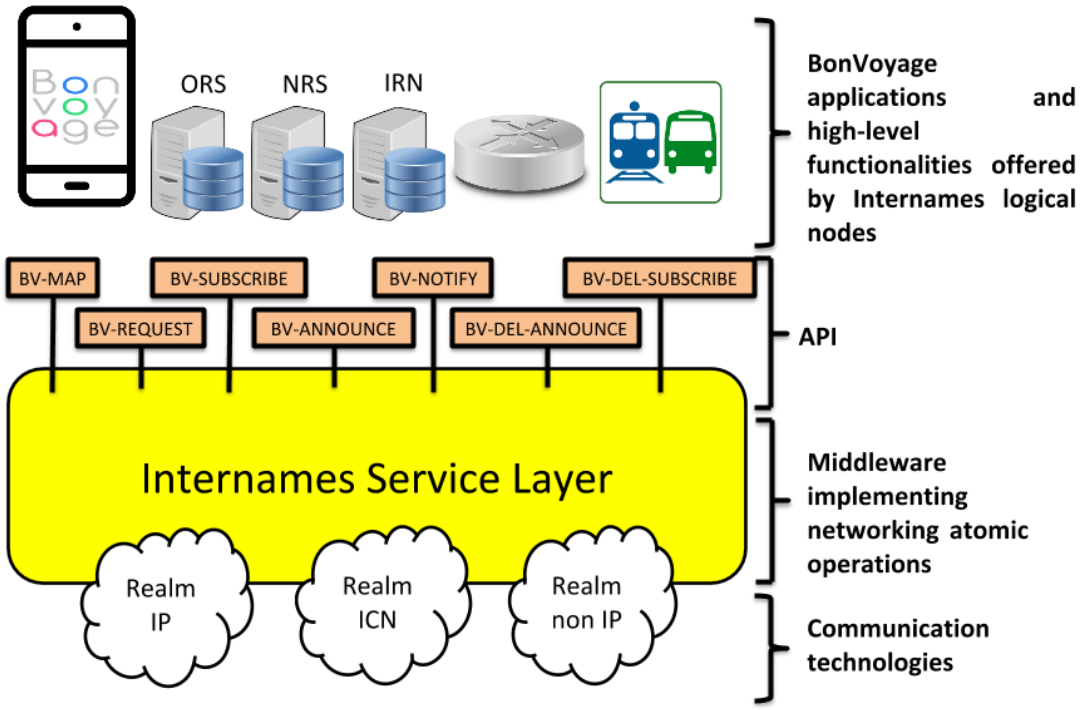

Figure 10: The Internames Service Layer and its APIs

4.2.1 Technical description of the Internames Service Layer

A more technical description of the Internames Service Layer is reported in Figure 11. It is possible to observe that it integrates three main modules: the *API manager* that offers the set of APIs to the application, the *networking manager* that implements atomic networking operations, and the *technology manager* that hosts technology-dependent instances able to process messages coming from the network realm (e.g., IP or ICN). The main rational characterizing the developed middleware is described in what follows:

- the high level applications interact with the API manager and it may use APIs for initiating some operations belonging to request-response or publish-subscribe communication schemes;
- each API triggers the execution of one or more atomic networking operations, handled by the networking manager;
- the commands generated by the networking manager is finally managed by the technology manager that translates them to concrete messages encoded according to the protocol adopted by the underlying communication technology;
- messages processed by the technology manager can be delivered to the application or to the networking manager.

The APIs offered by the Internames Service Layer are:

- **BV-MAP**: it triggers the mapping functionality offered by the ORS. As its name suggests, it is called when an input parameter, from the data consumer, is accepted and the research for a given content starts. As long as a query can be done with keywords and not only with the precise word, this API intervenes to resolve the potential ambiguity. Assuming meta-data or names are the used key by the data consumer, ORS responds with a list of all the names matching the request. Then, the function is asked to trigger the execution of the operations taking place within the overall system at a lower level, acting to coordinate functionalities. BV-MAP executes one single atomic networking operation, that is **low_bv_map.**
- **BV-REQUEST**: As said, requests are the beginning part of the communication scheme. This API receives an input parameter, that is the name that identifies the content or data of interest, and retrieves the result. The name can be the result of the execution of the BV-MAP API previously mentioned; for this reason, it is a precise reference selected between

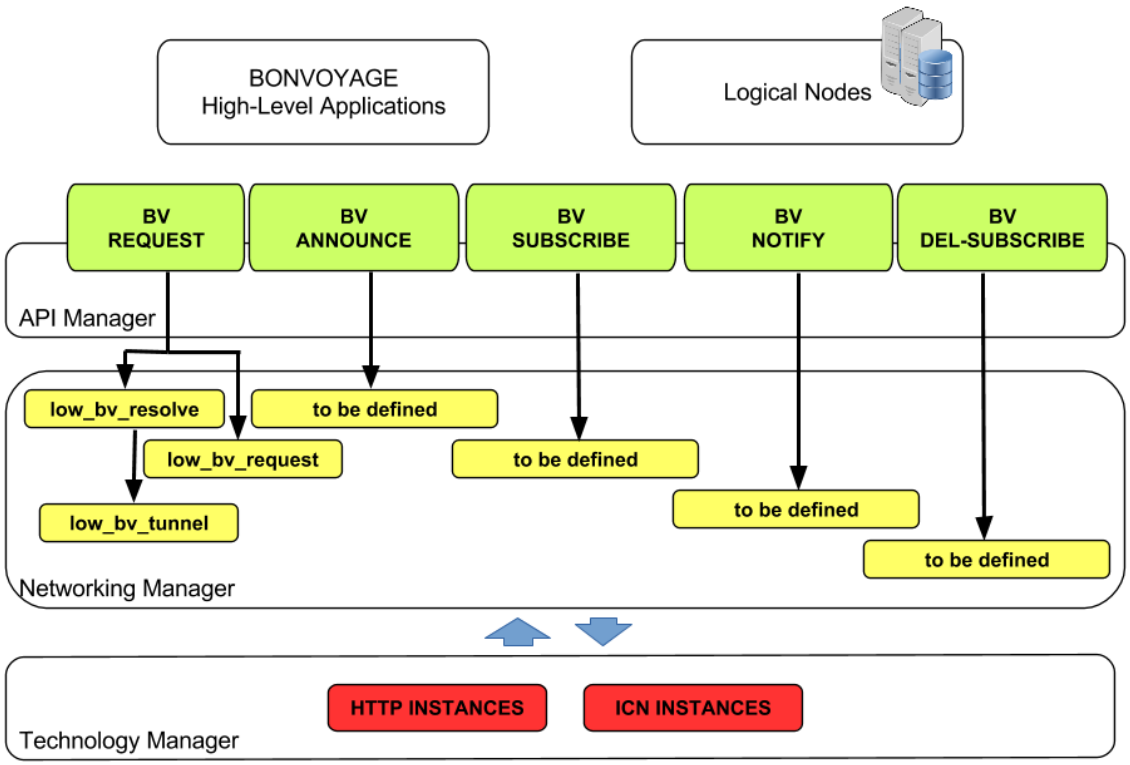

Figure 11: Technical details for the Internames Service Layer

the list of names that the data consumer has received. Since travel-centric and sensing contents can be retrieved from a different network realm (i.e., by establishing a multi-hop connection between data consumer and data producer), this functionality requires two additional operations to be performed, that are resolve and tunnel, in addition to the simple request operation. Both of them are defined at low level and are considered as atomic networking operations. In particular, if the node is directly connected to the realm specified in the name and the realm is based on the ICN protocol, the networking manager executes the *low_request* atomic networking operation and immediately sends the request by using ICN primitives; In case the node is directly connected to the realm specified in the name and the realm is based on the IP protocol, the networking manager contacts the NRS node for getting the IP address of the gateway that exposes that resource (this task is performed by *low_resolve*); then it executes the *low_request* atomic networking operation for sending the request by using IP primitives. Finally, if the node is not connected to the realm specified in the name, it is necessary to establish a multi-hop routing path across heterogeneous domains; to this end, the networking manager contacts the NRS node for getting routing information (this task is performed by *low_resolve*); then, *low_tunnel* is executed for delivering the request towards the next hop (a border router) identified by the NRS; the process is iterated until reaching the realm that hosts the data consumer.

- **BV-ANNOUNCE**: it will be described in Deliverable 3.2.
- **BV-DEL-ANNOUNCE**: it will be described in Deliverable 3.2.
- **BV-SUBSCRIBE**: it will be described in Deliverable 3.2.
- **BV-DEL-SUBSCRIBE**: it will be described in Deliverable 3.2.
- **BV-NOTIFY**: it will be described in Deliverable 3.2.

Figure 12 summarizes the details related to the APIs used for the request-response communication scheme.

It is important to note that the commands generated by the networking manager are translated by the technology manger according to the underlying technology. Further details have been summarized in Figure 13. For what concerns the IP realm, it is assumed that messages are exchanged by using the HTTP protocol. Without loss of generality, for the ICN realm, the Named Data Networking (NDN) is taken into account. In NDN, the communication is receiver-driven: contents are sent only in response to the respective requests, in a one-to-one relation; only two types of messages are admitted, that are *INTEREST* and *DATA*; an user may ask

API	Atomic networking functionalities triggered by the API	Description
BV-MAP	low_bv_map	**Role**: Provide content names and resolve potential ambiguity **Input**: names, keywords or search topic; **Output**: list of names matching the request
BV-REQUEST	low_bv_request low_bv_resolve low_bv_tunnel	**Role**: request a travel-centric information or participatory sensing data identified by a given name **Input**: name or identifier (output of BV-MAP); **Output**: location (next hop) and communication protocol

Figure 12: Atomic networking operations

for a content by issuing an *INTEREST* message, routed towards the nodes in possession of the required information; then, these nodes are triggered to reply with a corresponding *DATA* message (please, see the Appendix for more details).

API	Atomic Networking Operation	Realm	Message
BV-REQUEST	low_bv_request	IP	GET HTTP [IP ADDRESS]:[PORT]/[NAME]
		NDN	INTEREST /[NAME]
	low_bv_resolve	IP	GET HTTP [NRS_IP]:53/bv/resolve/n2n://[realm_ID]/[NAME]
		NDN	INTEREST /bv/resolve/n2n://[realm_ID]/NAME]
	low_bv_tunnel	IP	GET HTTP [NR_IP]:80/bv/tunnel/n2n://[realm_ID]/[NAME]
		NDN	INTEREST /bv/tunnel/[NR_ID]/n2n://[realm_ID]/[NAME]

Figure 13: Messages generated in IP and NDN network realms.

4.3 The namespace

Names are important because they are unique identifiers of data within the network. Until now, the way names are given and organized has been taken for granted. Now it is time for this information to be given and analyzed. The BONVOYAGE platform adopts a **realm-based**, **geo-referenced**, and **hierarchical** naming structure, useful to describe the availability of contents, devices, and services in a heterogeneous communication infrastructure, based on the geographical area they belong to.

4.3.1 Realm-based naming structure

The BONVOYAGE platform adopts a realm-based naming structure, useful to describe the availability of contents, devices, and services in a heterogeneous communication infrastructure:

n2n://[realm ID]/[NAME],

where *[realm ID]* is the network realm where the content is available and *[NAME]* is its unique name identifying travel-centric and sensing data, as well as control messages exchanged in the BONVOYAGE communication system during the provisioning of request-response and publish-subscribe services.

4.3.2 Geo-referenced names

The OpenGeoBase [20] approach implies a spatial indexing that uses a layered grid. The latter is aligned with world parallels and meridians. Grid regions are called tiles and they are different in size and every size implies a (refined) level. Tiles are aligned one with the other as they extend themselves starting from the same point. First-level tiles, namely level-0 tile, represent a grid containing world points having the same longitude and latitude values up to the dot (the whole part of the number), e.g. the level-0 tile (8,60). This tile contains all points with longitude and latitude starting from 8 and 60, respectively. As this indicates a specific but still large region, level-1 tile are used to include the first decimal of the two, respectively. A level-n tile of the grid contains all those points having the same latitude and longitude but in a smaller region. As the tiles grows in value (e.g. n= 1 or 2), value are considered up to the nth decimal, e.g. level-2 tile is 8.43, 61.56, so the selected region contains all points whose latitude and longitude start with 8.43 and 61.56, respectively. Resulting names are structured as follows:

[NAME] = /bv/lng(0)/lat(0)/lng(1)lat(1)/.../lng(n)lat(n)/GPS-ID/[other_info]

Without loss of generality, it is assumed that a tile has a geographical area of 10 km^2.

For instance, a content available in a geographical area having a tile described with GPS coordinates 8.43 and 61.56, is identified with the name:

[NAME] = /bv/61/8/54/63/GPS-ID/[other_info]

Note that the $[other\_info]$ field is used to specify further details, which depends from the specific service (more details will be provided in Deliverable 3.3 - Travel-centric and participatory sensing services).

4.3.3 Hierarchical naming

In hierarchical namespaces the names are composed by a succession of strings, optionally encrypted, in which every name prefix identifies a sub-tree in the namespace, giving native support for aggregation and multicasting capabilities. Hierarchical names are suitable for data that are frequently updated, hence suitable for highly dynamic contexts and real time applications. On the contrary, the variable-length name structure leads to both bandwidth and processing overheads. Each content is uniquely identified by a Content Name defined by a tree hierarchical structure. The BONVOYAGE platform adopts a hierarchical structure for the *[other_info]* part of the namespace, which specifies the standard which the name refers to, the provider of the named content, the specific requested service, and optionally additional infos. The resulting name is structured as follows:

$$\textbf{[other\_info]} = \textbf{/[std]/[provider\_id]/[service]/} < * >$$

For example,if the DatexII standard is used, the name appears in the form:

$$\textbf{[other\_info]} = \textbf{/datexii/npra/situation,}$$

Also in this case, more details will be provided in Deliverable 3.3.

4.3.4 Concrete Examples

Few concrete examples are provided in this section. The reference network architecture is reported in Figure 14.

In this network scenario a consumer is attached in an IP network with realm ID set to Net_IP_1. It is interested to retrieve the situation of traffic provided by NPRA (with DatexII standard) within the geographical area of 10 km^2 identified by the low-left corner GPS point (lat = 59.92, lon = 10.73). Thus, the consumer makes a request with the name:

n2n://[Net_IP_2]/bv/10/59/79/GPS-ID/datexii/npra/situation

Another consumer is deployed in a NDN network, with realm ID set to Net_NDN. It is interested to retrieve every information (i.e., not only situation provided by NPRA) within the 10 km^2 geographical area identified by the low-left corner GPS point (lat 60.36, lon 5.35). Therefore, it makes a request with the name:

n2n://[Net_IP_2]/bv/05/60/33/GPS-ID

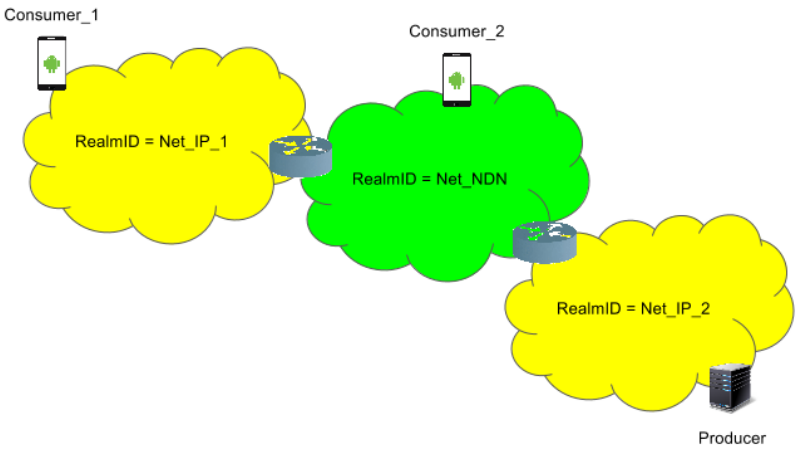

Figure 14: Network scenario used for the examples.

4.4 Final considerations on routing and caching algorithms

As already described in the previous sections, the NRS node, integrated in the BONVOYAGE Communication System, is in charge of generating routing paths between data consumer and data producer. To accomplish this task, is has to know the location of all the available data (i.e., travel-centric information and participatory sensing contents) within the heterogeneous network. These details are collected thanks to announce messages sent by data producers. Without loss of generality, in the BONVOYAGE project we plan to reuse consolidated routing strategies already proposed in the literature.

It is important to note that when the content is delivered to the data consumer, it may be cached in intermediate routers (mainly border routers connecting heterogeneous realms). For this reason, border routers become data producers for specific contents. Also in this case, the literature offers many caching algorithms to be reused in the BONVOYAGE project. Therefore, no novel approaches will be created in this context. However, there is an important issue to be properly investigated, that refers to the lifetime of a content itself. The question is: how long can a content be stored in a cache before being obsolete? The answer could be formulated only by investigating the properties of data produced in the BONVOYAGE platform and their frequency. These aspects will be investigated in Deliverable 3.3.

5 Routing operations triggered by the request-response communication scheme

The provisioning of request-response services within the BONVOYAGE Communication System is now explained through practical examples.

Without loss of generality, the considered scenario supposes that a data consumer (e.g., a journey operator or planner) is interested in getting information about an area of interest identified by the name:

n2n://[realm_ID]/[NAME],

where [NAME] is equals to:

[NAME] = bv/61/8/54/63/GPS-ID/datexii/npra/situation

At the network level, several configurations have been considered: they assume to integrate three different realms, which are IP, NDN, and PURSUIT. Note that the realm_IP refers to the current Internet; NDN and PURSUIT represent, instead, two concrete architectures based on the Information-Centric Networking (ICN) paradigm [10].

5.1 Messages exchanged in the IP Realm

In the IP realm, the messages are exchanged through the HyperText Transfer Protocol (HTTP). They are delivered by using the **TCP/IP protocol suite**, currently used in the Internet architecture.

To send a message to a given destination (that could be ORS, IRN, NRS, border router, data consumer, and data producer), it is necessary to explicitly state the IP address and the port number.

5.2 Messages exchanged in the NDN Realm

NDN natively supports a receiver-driven communication, according to which contents are sent by data producers only in response to the respective requests coming from data consumers. As default, data exchange follows the simple request-response communication model.

Only two types of messages are admitted: **INTEREST and DATA**. A user may ask for content by issuing an INTEREST, routed towards the nodes in possession of the required information. Then, these nodes are triggered to reply with DATA packets. The most important field of both INTEREST and DATA packets is the Content Name, which stores the name associated to the requested content.

5.3 Messages exchanged in the PURSUIT Realm

While IP and NDN realms natively support request-response services, data exchange in PURSUIT is based on the publish-subscribe mechanism. To solve this issue, a request-response scheme can be mapped to a couple of publish-subscribe sessions [21].

According to [21], in fact, contents are categorized in **scopes**. When a data consumer subscribes to a given scope, it implicitly announces its interest for all the contents that will be published under that scope.

Mapping, resolve, tunnel, announce, and subscription are atomic networking operations executed by the Internames Service Layer. They are identified within the following scopes: **MAP, RESOLVE, TUNNEL, ANNOUNCE, and SUBSCRIPTION**. Furthermore, for each of them, two sub-scopes are introduced too: **QUERY**, adopted to publish the request within a given scope, and **ANSWER**, reserved for publishing the corresponding response.

5.3.1 Mapping

The mapping functionality is used for retrieving names associated to a set of metadata. It provides the interaction between data consumer and ORS.

As illustrated in Figure 15, ORS firstly subscribes to QUERY under the scope MAP, thus being able to collect all the requests published under that scope. The data consumer publishes its request, i.e., the list of metadata, to the content Q, under the sub-scope QUERY. Then, data consumer immediately subscribes to the content A, under the sub-scope ANSWER, where the corresponding answer must be published to. Then, ORS publishes the response (which is, in this case, the set of names associated to the received metadata) in A. Finally, according to the PURSUIT architecture, the content will be delivered to the data consumer.

5.3.2 Resolve

The resolve functionality is one of the atomic networking operation associated to the routing procedure. It provides the interaction between data consumer and the reference NRS node.

As illustrated in Figure 16, NRS firstly subscribes to QUERY under the scope RESOLVE, thus being able to collect all the requests published under that scope. The data consumer publishes its request, i.e., the **name of the content to retrieve**, to the Q, under the sub-scope QUERY. Then, data consumer immediately subscribes to the content A, under the sub-scope ANSWER, where the corresponding answer must be published to. Then, NRS publishes the response (which is, in this case, **next hop and protocol type**) in A. Finally, according to the PURSUIT architecture, the content will be delivered to the data consumer.

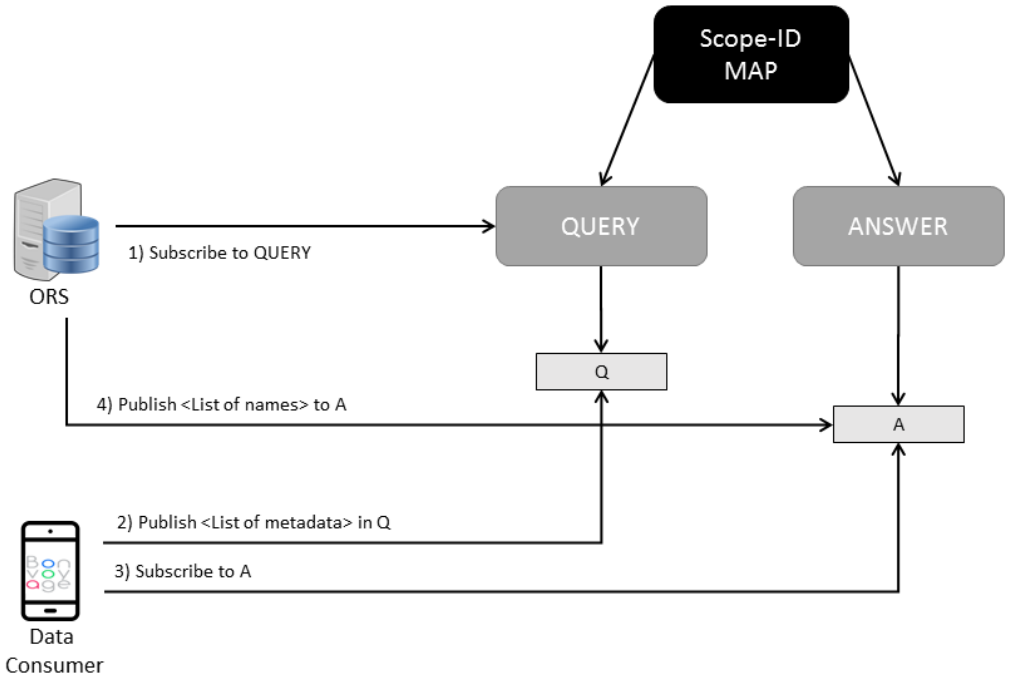

Figure 15: Provisioning of the mapping functionality in the PURSUIT realm

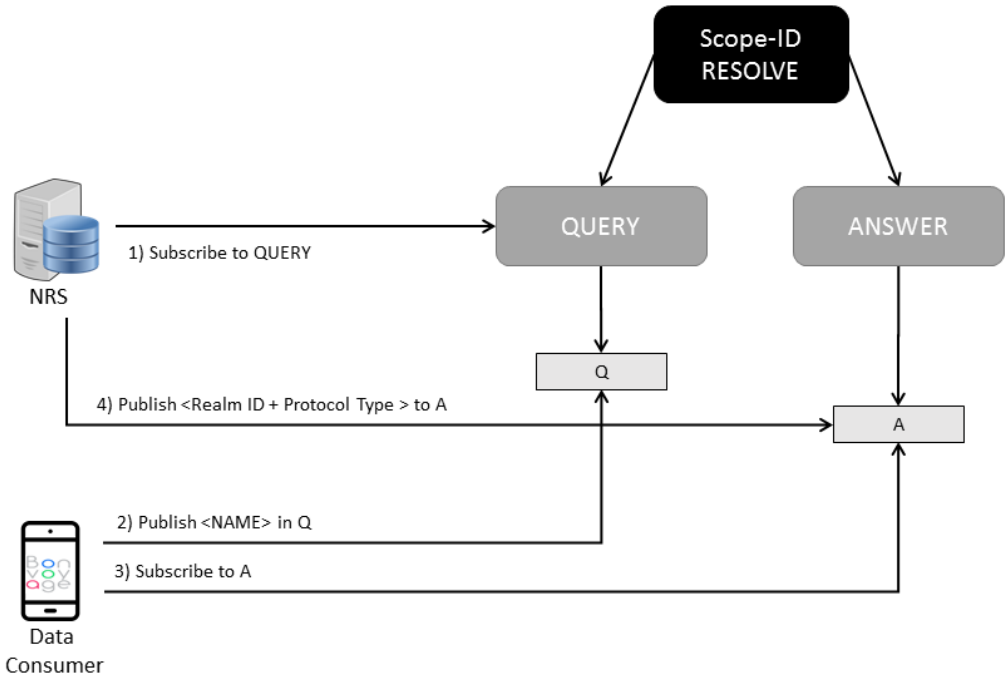

Figure 16: Provisioning of the resolve functionality in the PURSUIT realm

5.3.3 Tunnel

The tunnel functionality is one of the atomic networking operations associated to the routing procedure. It provides the interaction between data consumer and the border router identified with the aforementioned resolve functionality.

As illustrated in Figure 17, the border router (NR_1 in the proposed figure) firstly subscribes to QUERY under the scope TUNNEL, thus being able to collect all the requests published under that scope. The data consumer publishes its request, i.e., the **name of the content to retrieve**, to the content Q, under the sub-scope QUERY. Then, data consumer immediately subscribes to the content A, under the sub-scope ANSWER, where the corresponding answer must be published to. Then, the border router publishes the response (which is, in this case, **requested data**) in A. Finally, according to the PURSUIT architecture, the content will be delivered to the data consumer.

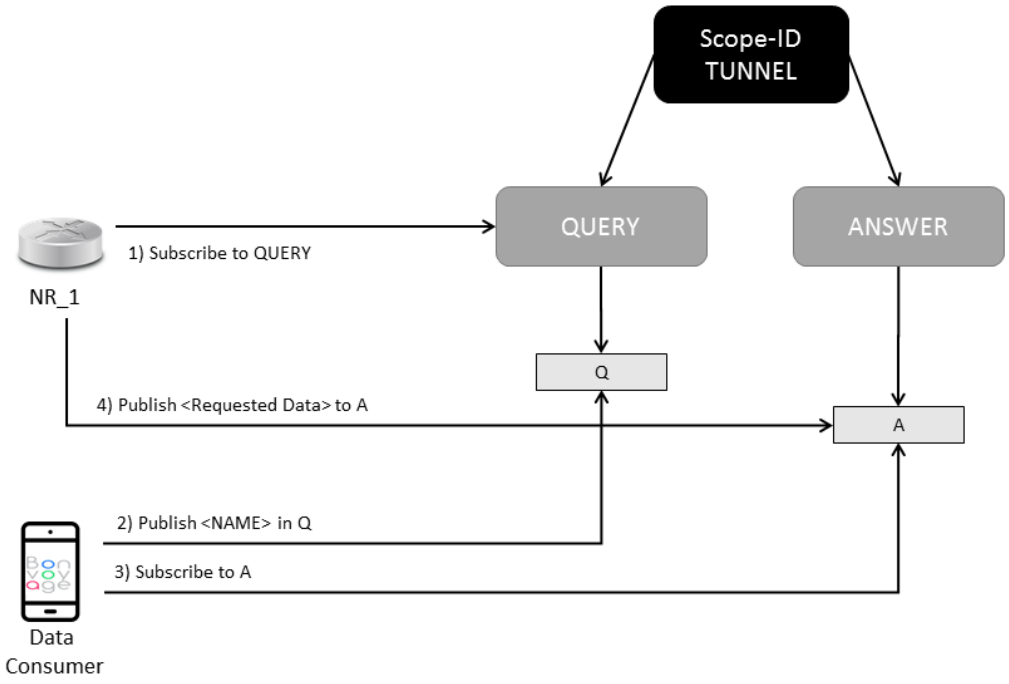

Figure 17: Provisioning of the tunnel functionality in the PURSUIT realm

5.4 IP-NDN Scenario

In this example, the data consumer is attached to the IP realm whereas data producer is attached to the NDN realm. The travel-centric data is identified with the name:

n2n://[realm_NDN]/[NAME]

The message sequence chart describing the interaction among entities in the BONVOYAGE platform is shown in Figure 18. The request-response communication scheme involves the following steps:

1. The data consumer has to discover the name of the content it is willing to retrieve. To this end, it uses the BV-MAP API to issue a mapping request to the reference ORS of its network realm. The Internames Service Layer executes the low_bv_map atomic operation and translates this high-level request in a GET HTTP:

   HTTP: GET

   [ORS_IP]:80/bv/map/⟨list of metadata⟩

   that will be sent to the ORS logical node.

2. ORS answers with a list of names identifying requested contents.

3. The data consumer selects a content (or more) of its interest. In our example it is:

   n2n://[realm_NDN]/[NAME]

   and initiates the request-response data exchange through the BV-REQUEST API.

4. The Internames Service Layer instance of the data consumer executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends a GET HTTP message to the reference NRS instance:

   HTTP: GET

   [NRS_IP]:53/bv/resolve/n2n://[realm_NDN]/[NAME]

5. The NRS generates a corresponding answer, highlighting that the request should be forwarded to the router NR_1, by using the IP protocol.

6. As a consequence, the Internames Service Layer instance running on the data consumer executes the low-bv-tunnel atomic networking operation that forwards the request to the router NR_1. The message is sent through the IP protocol by using a GET HTTP message:

HTTP: GET

[NR_1]:80/bv/tunnel/n2n://[realm_NDN]/[NAME]

7. The message is processed by the router, that immediately realizes that the requested content is available in the NDN realm to which it is directly attached. Therefore, it extracts from the realm-based naming structure the part of the name to use within the NDN realm, i.e.,

NDN: INTEREST

/[NAME]

and sends an INTEREST packet in NDN realm.

8. The data producer answers with a corresponding DATA packet storing the requested travel-centric data.

9. The router processes the DATA packet and forwards the received content in the IP realm, back to the data consumer.

5.5 NDN-IP Scenario

In this example, the data consumer is attached to the NDN realm whereas the data producer is attached to the IP realm. The travel-centric data is identified by the name:

n2n://[realm_IP]/[NAME]

The message sequence chart describing the interaction among entities in the BONVOYAGE platform is shown in Figure 19. The request-response communication scheme involves the following steps:

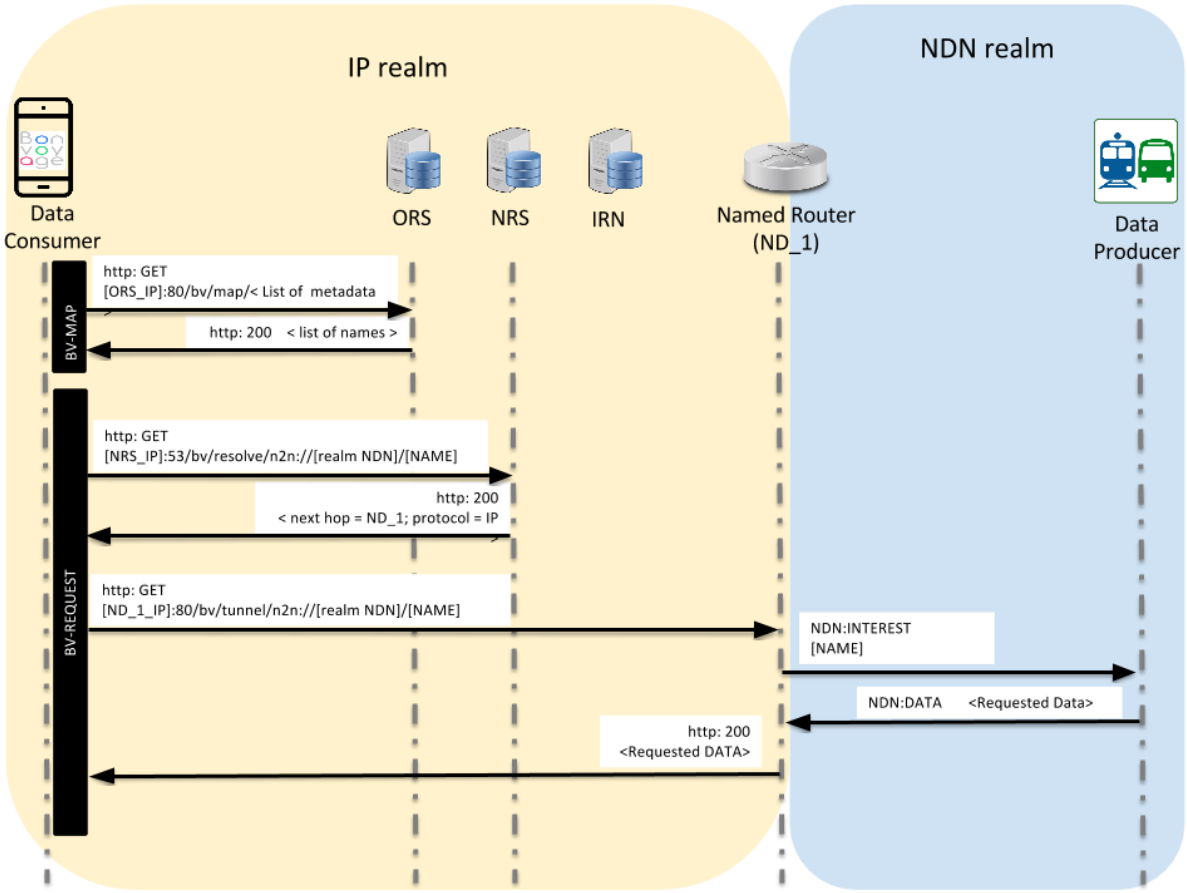

Figure 18: Provisioning of request-response services in the IP-NDN scenario

1. The data consumer has to discover the name of the content it is willing to retrieve. To this end, it uses the BV-MAP API to issue a mapping request to the reference ORS of its network realm. The Internames Service Layer executes the low_bv_map atomic operation and translates this high-level request to a NDN INTEREST:

NDN: INTEREST

/bv/map/⟨List of metadata⟩

that will be sent to the ORS logical node.

2. ORS answers with a list of names identifying requested contents.

3. The data consumer selects a content (or more) of its interest. In our example is:

n2n://[realm_IP]/[NAME]

Then it initiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends an INTEREST message to the reference NRS node:

NDN: INTEREST

/bv/resolve/n2n://[realm_IP]/[NAME]

4. The NRS generates a corresponding answer, highlighting that the request should be forwarded to the router **NR_1**, by using the **NDN protocol**.

5. As a consequence, the Internames Service Layer instance running on the data consumer executes the low-bv-tunnel atomic networking operation that forwards the request to the router NR_1. The message is sent through the NDN protocol by using a NDN INTEREST message:

NDN: INTEREST

/bv/tunnel/[NR_1]/n2n://[realm_IP]/[NAME]

6. The message is processed by the router, that reinitiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends a GET HTTP to the reference NRS node:

HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_IP]/[NAME]

7. The reference NRS node answers with routing details:

HTTP: 200

⟨next hop=[producer_IP]:[port]; protocol=IP⟩

8. At this point, the Internames Service Layer instance realizes that the requested content is available in the IP realm to which it is directly attached. Therefore, it extracts from the realm-based naming structure [NAME] to use within the IP realm and sends a GET HTTP towards the IP address of the data producer:

HTTP: GET

[producer_IP]:[port]/[NAME]

9. The data producer answers with the requested travel-centric data.

10. The router NR_1 processes the messages and stores the received content within a DATA packet to forward information in the NDN realm, back to the data consumer.

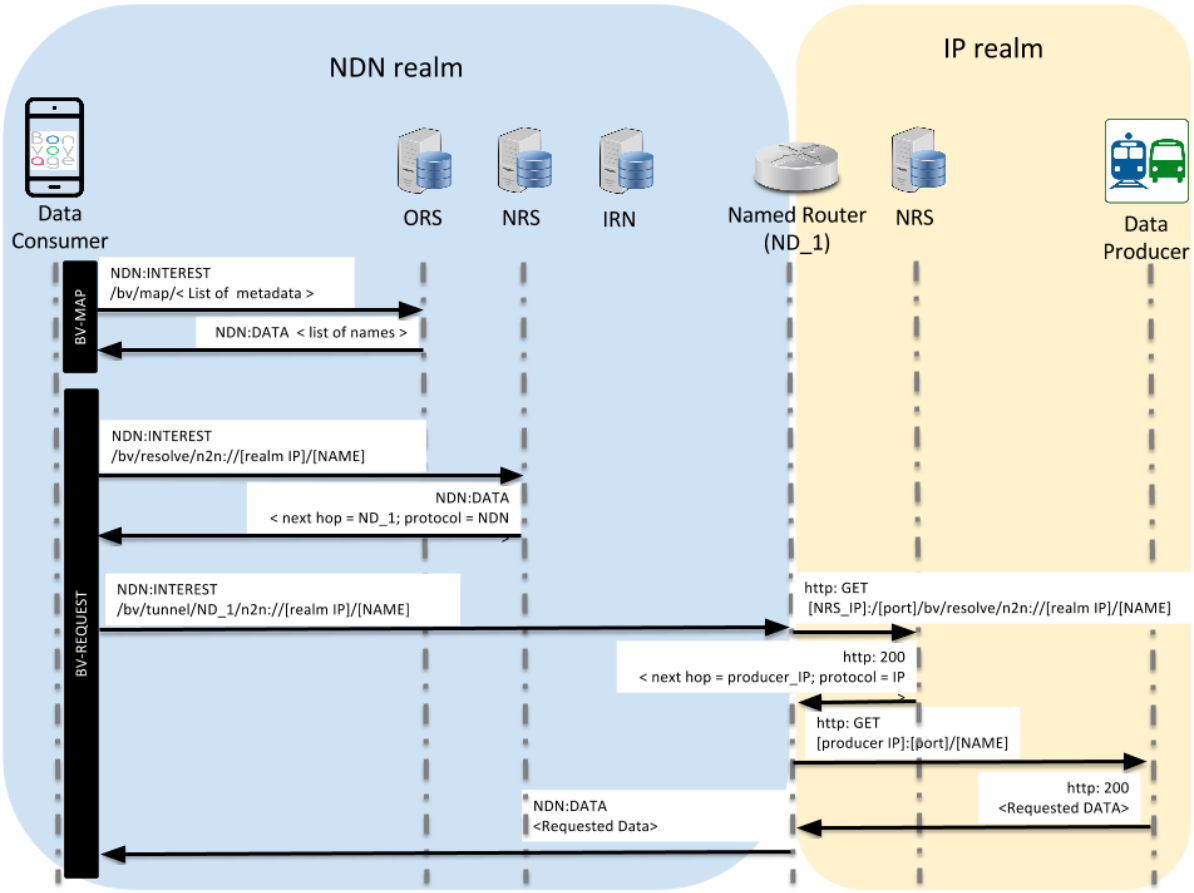

Figure 19: Provisioning of request-response services in NDN-IP scenario

5.6 IP-NDN-IP Scenario

In this example, both data consumer and data producers are attached to two different IP realms, connected to each other through a NDN realm. The travel-centric data is identied with the name:

n2n://[realm_IP]/[NAME]

The message sequence chart in Figure 20 describes a more complex scenario in which communication is executed among three different realms. The request-response communication scheme involves the the following steps:

1. The data consumer has to discover the name of the content it is willing to retrieve. To this end, it uses the BV-MAP API to issue a mapping request to the reference ORS of its network realm. The Internames Service Layer executes the low_bv_map atomic operation and translates this high-level request to a GET HTTP:

   MSG_1: HTTP: GET

   [ORS_IP]:80/bv/map/⟨list of metadata⟩

   that will be sent to the ORS logical node.

2. ORS answers with a list of names identifying requested contents

   MSG_2: HTTP: 200

   ⟨list of names⟩

3. The data consumer selects a content of its interest. In our example it is:

   n2n://[realm_IP]/[NAME]

   and initiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends a GET HTTP to the reference NRS node:

   MSG_3: HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_IP]/[NAME]

4. The reference NRS node answers with routing details:

MSG_4: HTTP: 200

⟨next hop=NR_1; protocol=IP⟩

5. The Internames Service Layer instance of the data consumer executes the low-bv-tunnel atomic networking operation that forwards the request to the router NR_1. To this end, it sends a GET HTTP message to the reference NRS node:

MSG_5: HTTP: GET

[NR_1]:80/bv/tunnel/n2n://[realm_IP]/[NAME]

6. NR_1 asks the NRS within NDN realm how to resolve the name of the content of interest for the data consumer.

MSG_6: NDN: INTEREST

/bv/resolve/n2n://[realm_IP]/[NAME]

7. The answer comes from the NRS within the NDN realm and indicates the following hop to NR_2, the router between NDN and IP realms:

MSG_7: NDN: DATA

⟨next hop=NR_2; Protocol = NDN⟩

8. NR_1 delivers the request to NR_2 through an INTEREST message:

MSG_8: NDN: INTEREST

/bv/tunnel/NR_2/n2n://[realm_IP]/[NAME]

9. The message is processed by the router, that reinitiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends a GET HTTP to the reference NRS node:

MSG_9: HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_IP]/[NAME]

10. The reference NRS node answers with routing details:

MSG_10: HTTP: 200

⟨next hop=[producer_IP]:[port]; protocol=IP⟩

11. At this point, the Internames Service Layer instance realizes that the requested content is available in the IP realm to which it is directly attached. Therefore, it extracts from the realm-based naming structure [NAME] to use within the IP realm and sends a GET HTTP towards the IP address of the data producer:

MSG_11: HTTP: GET

[producer_IP]:[port]/[NAME]

12. The data producer answers with the requested travel-centric data.

MSG_12: HTTP: 200

⟨Requested Data⟩

13. The router NR_2 processes the messages and stores the received content within a DATA packet to forward in the NDN realm, back to the data consumer.

MSG_13: HTTP: 200

⟨Requested Data⟩

14. The router NR_1 processes the messages and stores the received content within HTTP message to forward in the IP realm, back to the data consumer.

MSG_14: NDN: DATA

⟨Requested Data⟩

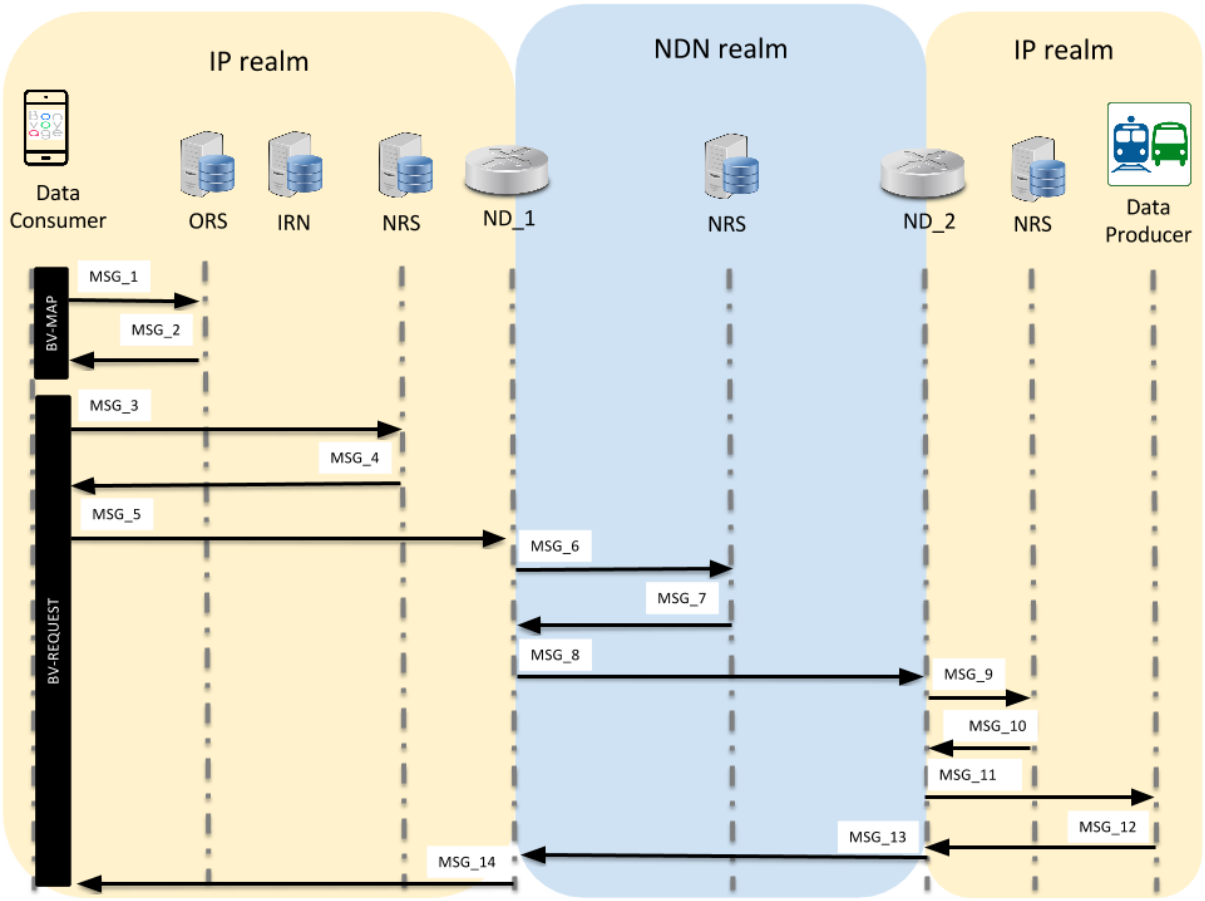

Figure 20: Provisioning of request-response services in IP-NDN-IP scenario

5.7 NDN-IP-NDN Scenario

In this example, both data consumer and data producers are attached to two different NDN realms, connected each other through an IP realm. The travel-centric data is identified by the name:

n2n://[realm_NDN]/[NAME]

In that case, the request-response communication scheme involves the following steps (see Figure 21):

1. The data consumer has to discover the name of the content it is willing to retrieve. To this end, it uses the BV-MAP API to issue a mapping request to the reference ORS of its network realm. The Internames Service Layer executes the low_bv_map atomic operation and translates this high-level request to a GET HTTP:

MSG_1: NDN: INTEREST

/bv/map/⟨list of metadata⟩

that will be sent to the ORS logical node.

2. ORS answers with a list of names identifying requested contents

MSG_2: NDN: DATA

⟨list of names⟩

3. The data consumer selects a content of its interest. In our example it is:

n2n://[realm_NDN]/[NAME]

and initiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends an INTEREST message to the reference NRS node:

MSG_3: NDN: INTEREST

/bv/resolve/[NAME]

4. The reference NRS node answers with routing details:

MSG_4: HTTP: 200

⟨next hop=NR_1; protocol=IP⟩

5. The Internames Service Layer instance running on the data consumer executes the low-bv-tunnel atomic networking operation that forwards the request to the router NR_1. The message is sent through the NDN protocol, by using an INTEREST message:

MSG_5: NDN: INTEREST

/bv/tunnel/NR_1/[NAME]

6. NR_1 contacts the NRS deployed in the IP realm for getting further routing information:

MSG_6: HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_NDN]/[NAME]

7. The NRS answers with routing details:

MSG_7: HTTP: 200

⟨next hop=NR_2; Protocol = IP⟩

8. NR_1 delivers the request to NR_2 through a GET HTTP message:

MSG_8: HTTP: GET

[NR_2_IP]:80/bv/tunnel/n2n://[realm_NDN]/[NAME]

9. The message is processed by the router, that immediately realizes that the requested content is available in the NDN realm to which it is directly attached. Therefore, it extracts from the realm-based naming structure the part of the name to use within the NDN realm and sends an INTEREST packet in the NDN realm:

MSG_9: NDN: INTEREST

/[NAME]

10. The data producer answers with a corresponding DATA packet storing the requested travel-centric data.

MSG_10: NDN: DATA

⟨Requested Data⟩

11. The router NR_2 processes the DATA packet and forwards the received content to NR_1 through a HTTP message:

MSG_11: HTTP: 200

⟨Requested Data⟩

12. The router NR_1 processes the HTTP message forwards the received content using a NDN DATA packet, back to the data consumer:

MSG_12: NDN: DATA

⟨Requested Data⟩

5.8 IP-PURSUIT-NDN Scenario

It is now assumed that data consumer and data producer are attached to IP and NDN realms, respectively. However, in this example, things get a little more complicated when communication has to be forwarded between three different realms if PURSUIT is the one in the middle. The travel-centric data is identified by the name:

n2n://[realm_NDN]/[NAME]

The request-response communication scheme involves the following steps:

1. The data consumer has to discover the name of the content it is willing to retrieve. To this end, it uses the BV-MAP API to issue a mapping request to the reference ORS of its

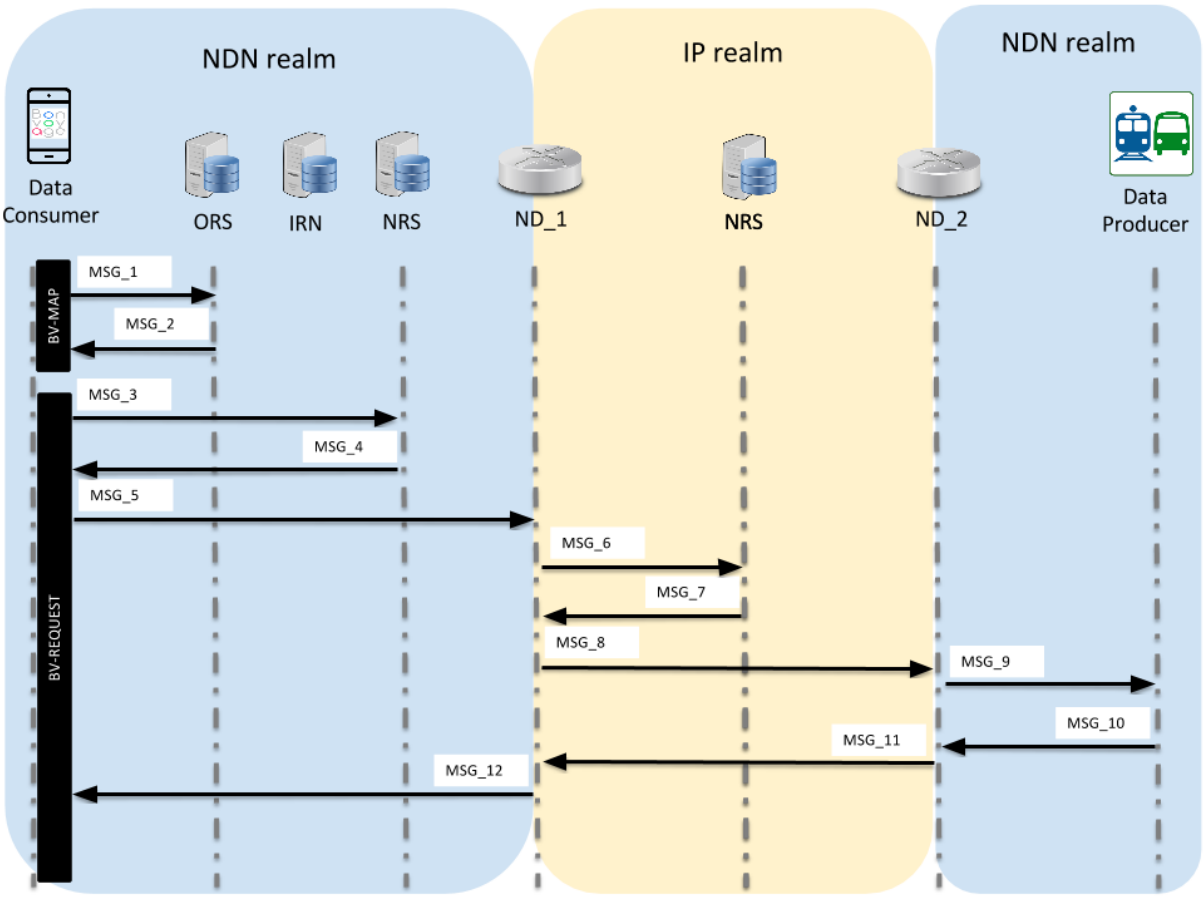

Figure 21: Provisioning of request-response services in NDN-IP-NDN scenario

network realm. The Internames Service Layer executes the low_bv_map atomic operation and translates this high-level request to a GET HTTP:

MSG_1:HTTP: GET

[ORS_IP]:80/bv/map/⟨list of metadata⟩

that will be sent to the ORS logical node.

2. ORS answers with a list of names identifying requested contents

MSG_2: HTTP: 200

⟨list of names⟩

3. The data consumer selects a content of its interest. In our example it is:

n2n://[realm_NDN]/[NAME]

and initiates the request-response data exchange through the BV-REQUEST API. Thus, the Internames Service Layer instance executes the low-bv-resolve atomic networking operation for getting the next-hop towards which sending the request. To this end, it sends a GET HTTP to the reference NRS node:

MSG_3: HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_NDN]/[NAME]

4. The reference NRS node answers with routing details:

MSG_4: HTTP: 200

⟨next hop=NR_1; protocol=IP⟩

5. The Internames Service Layer instance of the data consumer executes the low-bv-tunnel atomic networking operation that forwards the request to the router NR_1. To this end, it sends a GET HTTP message to the reference NRS node:

MSG_5: HTTP: GET

[NR_1]:80/bv/tunnel/n2n://[realm_NDN]/[NAME]

6. NR_1 contacts the NRS node for retrieving further routing information:

MSG_6: HTTP: GET

[NRS_IP]:53/bv/resolve/n2n://[realm_NDN]/[NAME]

7. The reference NRS node answers with routing details:

MSG_7: HTTP: 200

⟨next hop=NR_2; protocol=PURSUIT⟩

8. With reference to the PURSUIT realm, it is assumed that NR_2 is subscribed to the sub-scope QUERY of the topic TUNNEL and that NR_1 is subscribed to the sub-scope ANSWER of the topic TUNNEL. Therefore, NR_1 publishes the request under the sub-scope QUERY of the topic TUNNEL.

9. NR_2 receives the query according to the PURSUIT architecture. Then, it realizes that the requested content is available in the NDN realm to which it is directly connected. Therefore, it forwards the request through a NDN INTEREST:

MSG_8: NDN: INTEREST

/[NAME]

10. The data producer provides the requested travel-centric data.

MSG_9: NDN: DATA

⟨Requested Data⟩

11. NR_2 processes the messages and publishes the received content within the sub-scope ANSWER of the topic TUNNEL.

12. NR_1 receives the content according to the PURSUIT architecture and forwards it back to the data consumer.

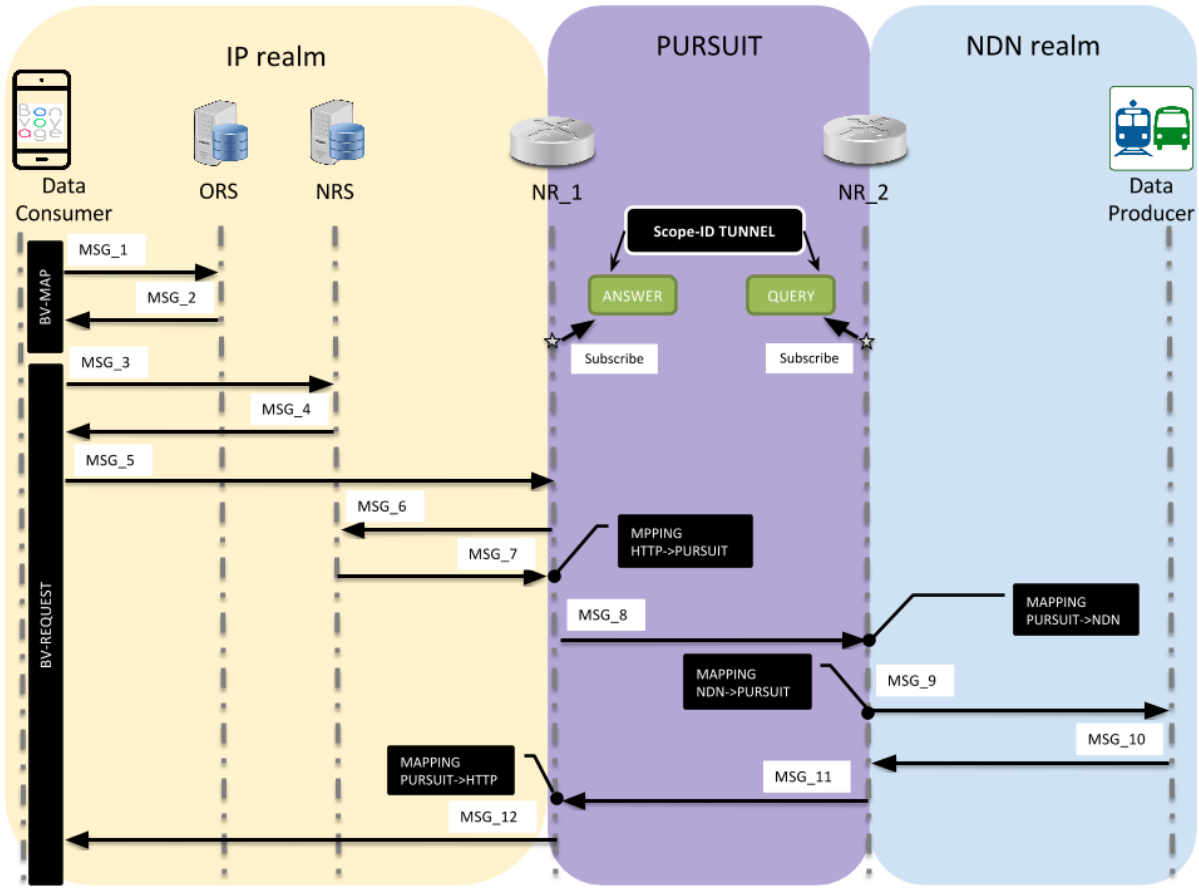

Figure 22: Provisioning of request-response services in IP-PURSUIT-NDN scenario

6 Preliminary implementation of the experimental testbed

The main functionalities of the BONVOYAGE Communication System are explored hereby through a preliminary experimental testbed.

The experimental testbed aims at demonstrating that a consumer application is able to retrieve travel-centric data in the BONVOYAGE Communication System platform by only using the communication functionalities exposed by the Internames Service layer (i.e., the developed middleware), thus completely ignoring all the details about the underlying technologies, the heterogeneous nature of the entire network, and the specific protocol architecture implemented in the network domains.

The implemented scenario is depicted in Figure 23. At the network layer, there are three network realms: **Net_1**,and **Net_2**, and **polibaNet**. While **Net_1** and **polibaNet** are based on the conventional TCP/IP protocol stack (e.g., used in the current Internet), **Net_2** is based on the NDN protocol. These realms are connected through border routers, namely **NR_ID_1** and **NR_ID_2**, that also hosts the instances of **NRS** and **IRN** logical nodes.

Consumer proxy and **producer proxy** offer a simplified interface for both consumer and producer. From one side, the **consumer proxy** receives requests from the consumer and retrieves asked data by using high-level APIs offered by the **Internames Service Layer**. From another side, the **producer proxy** collects data from external servers and exposes them according to the realm-based, geo-referenced, and hierarchical namespace properly designed for BONVOYAGE. The implementation of both consumer and producer proxies are strictly related to the travel-centric service to be offered. Some novel solutions will be carefully described in Deliverable 3.3. In what follows, instead, only networking-level aspects will be highlighted. Thus, without loss of generality and for sake of simplicity, we assume in this simple example that the data consumer is interested in retrieving travel-centric data (i.e., *Datex II* situations provided by NPRA and properly processed by the producer proxy) that refers to these two geographical areas: (1) a first tile of $10km^2$, having a low-left GPS point with coordinates $lat = 58.8$ and $lon = 7.1$ and (2) a second tile of $10km^2$, having a low-left GPS point with coordinates $lat = 58.8$ and $lon = 7.2$. Contents are identified by the names:

n2n://polibaNet/bv/07/58/1/8/GPS-ID/datexII/npra/situation,

n2n://polibaNet/bv/07/58/2/8/GPS-ID/datexII/npra/situation.

From the sketch of the deployed testbed it emerges that the **Internames Service Layer** is installed on consumer proxy, producer proxy, border routers, and logical nodes of Internames.

The sequence diagram related to the the data retrieving process is reported in Figure 24.

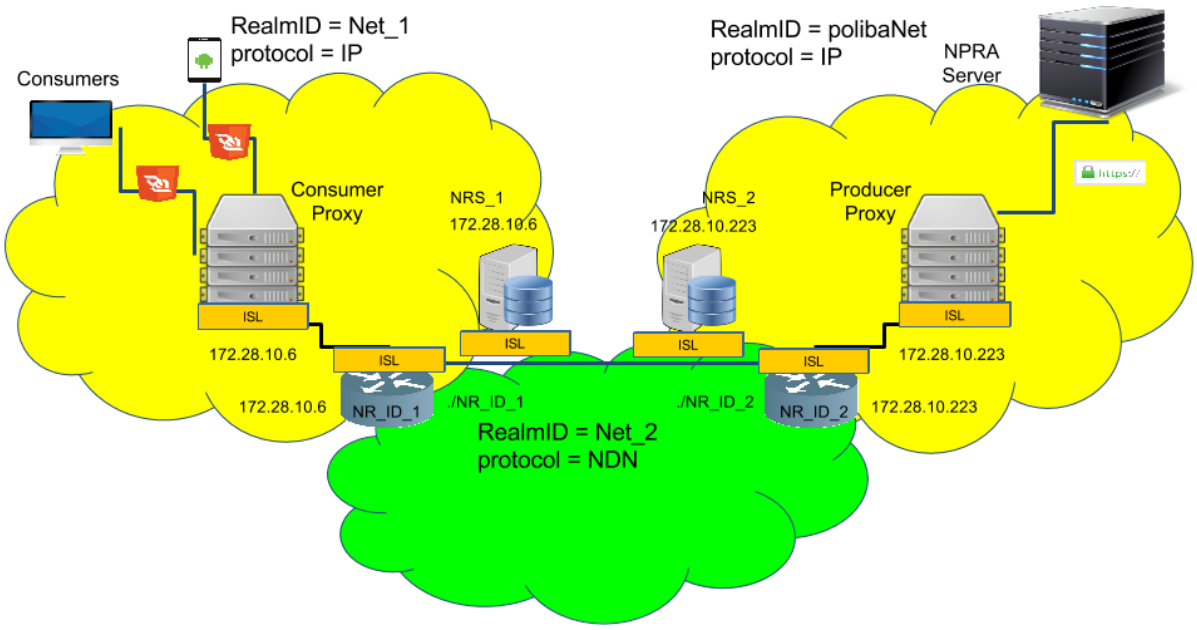

Figure 23: Big picture of the deployed experimental testbed.

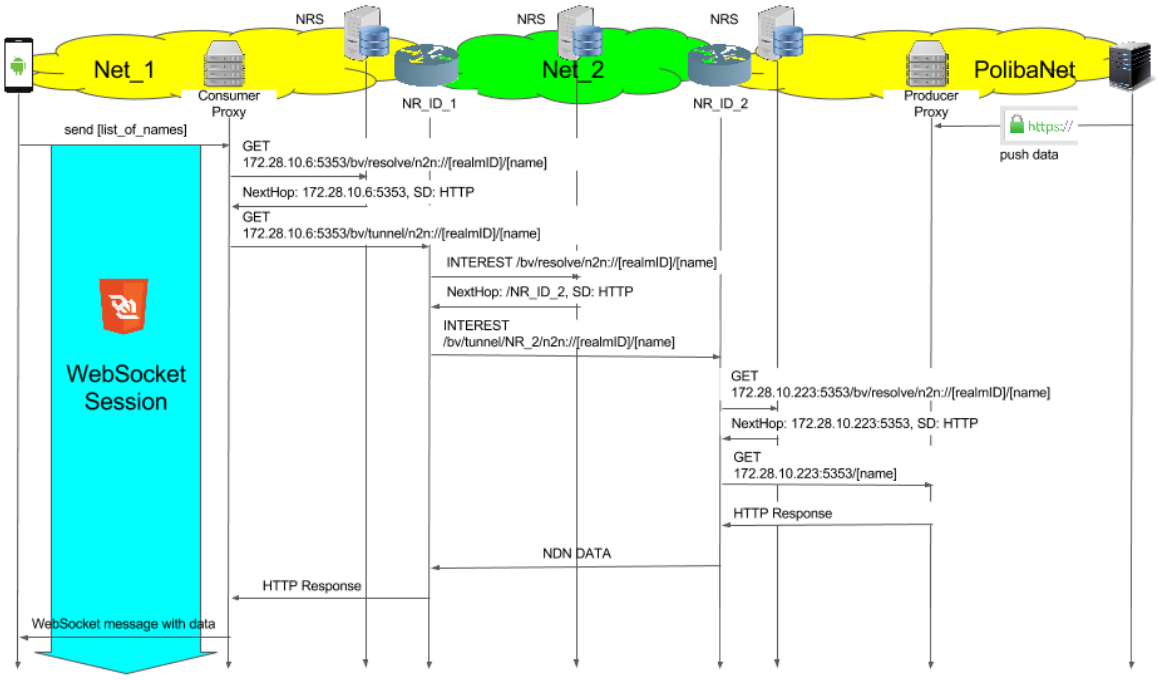

Figure 24: Sequence diagram.

The entire process involves the following steps:

- The **NR_ID_1** initiates the **/bv/tunnel** service. It is attached to both IP and NDN realms. The technology manager of the designed middleware hosts a HTTP server (IP address set to 172.28.10.6, port 8084) for the IP realm and a NDN instance for the NDN realm. The NDN instance registers the prefix **/bv/tunnel/NR_ID_1/** in the NDN realm, thus being able to receive any request belonging to the tunnel service (i.e., inter-realm routing).

- The **NR_ID_2** initiates the **/bv/tunnel** service. It is attached to both IP and NDN realms. The technology manager of the designed middleware hosts a HTTP server (IP address set to 172.28.10.223, port 8084) for the IP realm and a NDN instance for the NDN realm. The NDN instance registers the prefix **/bv/tunnel/NR_ID_2/** in the NDN realm, thus being able to receive any request belonging to the tunnel service (i.e., inter-realm routing).

- **NRS_1** initiates the **/bv/resolve** service. It is attached to both IP and NDN realms. The technology manager of the designed middleware hosts a HTTP server (IP address set to 172.28.10.6, port 5353) for the IP realm and a NDN instance of the NDN realm. The NDN instance registers the prefixes **/bv/resolve** in the NDN realm, thus being able to receive any request belonging to the resolve service.

- **NRS_2** initiates the **/bv/resolve** service. It is attached to both IP and NDN realms. The technology manager of the designed middleware hosts a HTTP server (IP address set to 172.28.10.223, port 5353) for the IP realm and a NDN instance of the NDN realm. The NDN instance registers the prefixes **/bv/resolve** in the NDN realm, thus being able to receive any request belonging to the resolve service.

- The **consumer** is implemented through a JAVA software, which initiates a web socket with the **consumer proxy**, sends the requests and collects the corresponding answers. Figure 25 shows the startup of the consumer.

- The **consumer proxy** receives and processes the requests sent by the consumer. Thus, it must now retrieve contents from the BONVOYAGE platform on behalf of the consumer. To this end, it uses the **BV-REQUEST API** to retrieve data from the BONVOYAGE Communication System through through the Internames Service Layer. This high-level API triggers the networking manager to execute the **low_resolve** atomic operation and

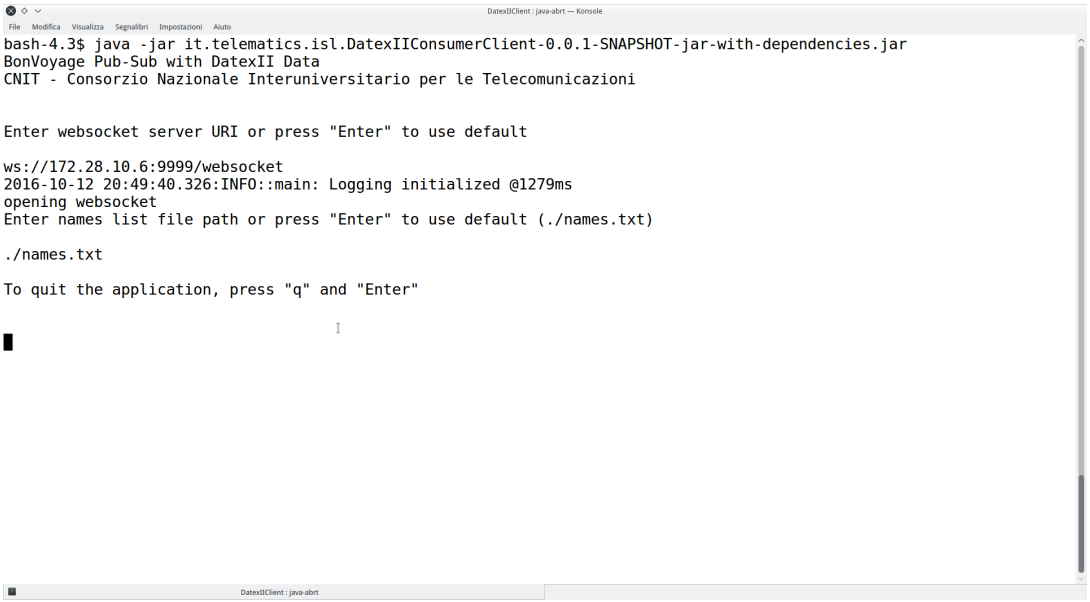

Figure 25: Consumer startup.

the technology manager to generate a GET HTTP message to sent to the reference NRS node.

- **NRS_1** answers by suggesting to send the request towards the border router **NR_ID_1** by using the HTTP protocol. Figures 26 and 27 report the logs related to the set of operations executed by the logical node **NRS_1**.
- The networking manager of the middleware instance running on the **consumer proxy** executes the **low_tunnel** atomic operation. Therefore, the technology manager sends a GET HTTP message to **NR_ID_1**, asking for the contents requested by the consumer.
- **NR_ID_1** processes the message. It realizes that it has to retrieve the content form the BONVOYAGE platform. To this end, it uses the **BV-REQUEST API** to retrieve data from the BONVOYAGE Communication System through through the Internames Service Layer. This high-level API triggers the networking manager to execute the **low_resolve** atomic operation and the technology manager to generate a GET HTTP message to sent to the reference NRS node.
- **NRS_1** answers by suggesting to send the request towards the border router **NR_ID_2** by using the NDN protocol. Figures 26 and 27, already introduced before, report the logs related to the set of operations executed by the logical node **NRS_1**.
- The networking manager of the middleware instance running on the **NR_ID_1** executes the **low_tunnel** atomic operation. Therefore, the technology manager sends a **NDN**

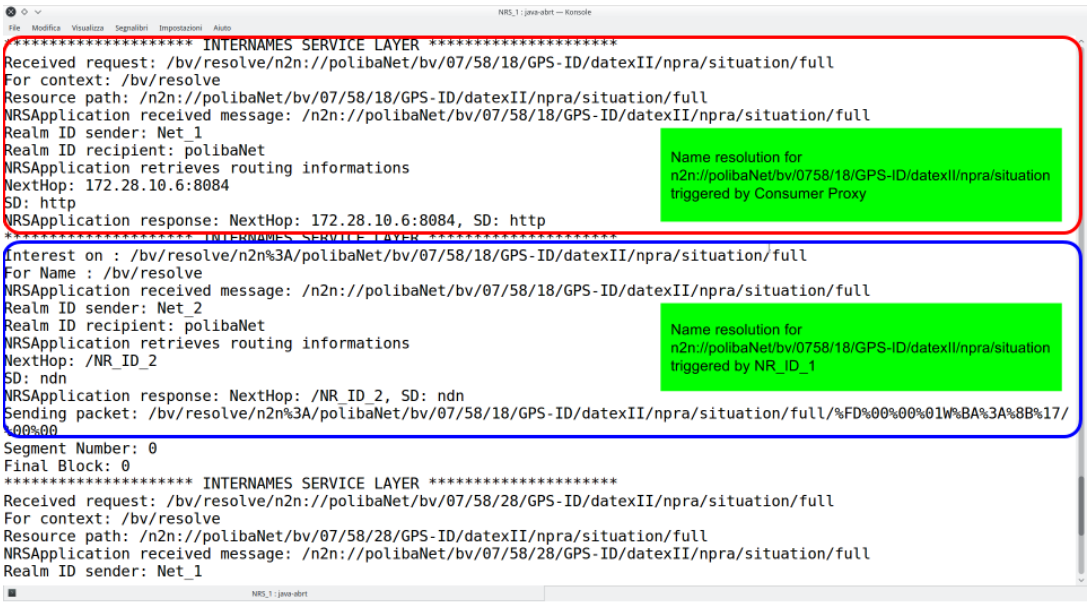

Figure 26: Logs reporting the set of operations executed by NRS_1 during the resolution process for the first name.

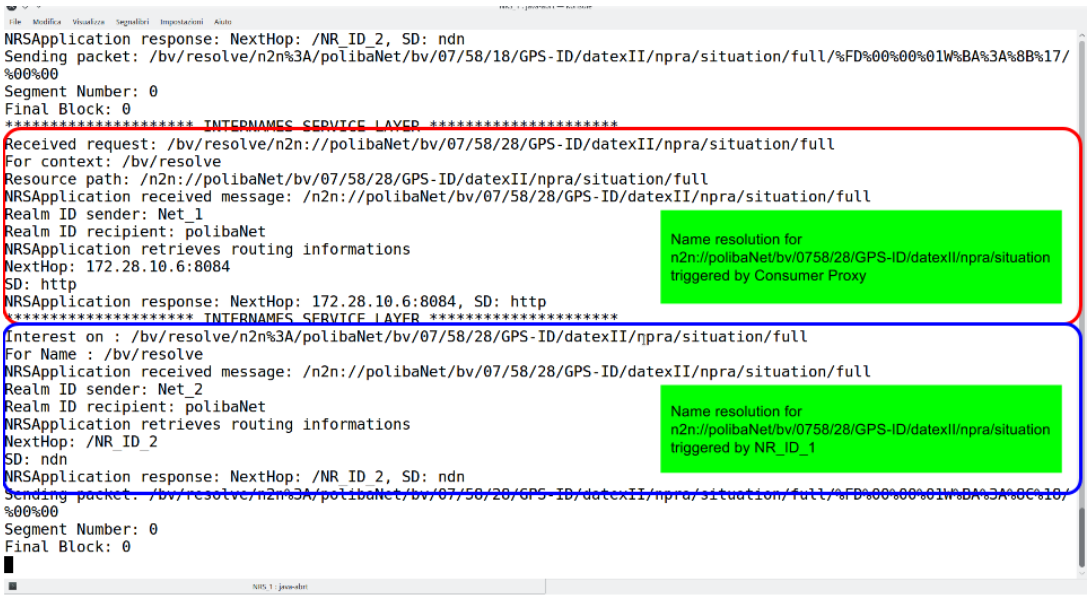

Figure 27: Logs reporting the set of operations executed by NRS_1 during the resolution process for the second name.

INTEREST message to **NR_ID_2**, asking for the contents requested by the consumer. Figure 28 reports the logs related to the set of operations executed by the logical node **NR_ID_1**.

Figure 28: Logs reporting the set of operations executed by NR_ID_1 during the routing process.

- **NR_ID_2** processes the message. It realizes that it has to retrieve the content form the BONVOYAGE platform. To this end, it uses the **BV-REQUEST API** to retrieve data from the BONVOYAGE Communication System through through the Internames Service Layer. This high-level API triggers the networking manager to execute the **low_-resolve** atomic operation and the technology manager to generate a GET HTTP message to sent to the reference NRS node. Figures 30 and 31 report the logs related to the set of operations executed by the logical node **NRS_ID_2**.

- **NRS_2** answers by indicating the IP address of the **producer proxy** that stores the requested data. Figure 29 reports the logs related to the set of operations executed by the logical node **NRS_2**.

- The networking manager of the middleware instance running on the **NR_ID_2** immediately execute the **low_request** atomic operation. Therefore, the technology manager sends a **GET HTTP** message to **producer proxy**, asking for the contents requested by the consumer. Figures 30 and 31, already introduced before, report the logs related to the set of operations executed by the logical node **NR_ID_2**.

- Once the request is received by the **producer proxy**, a new data is generated and sent

Figure 29: Logs reporting the set of operations executed by NRS_2 during the resolution process.

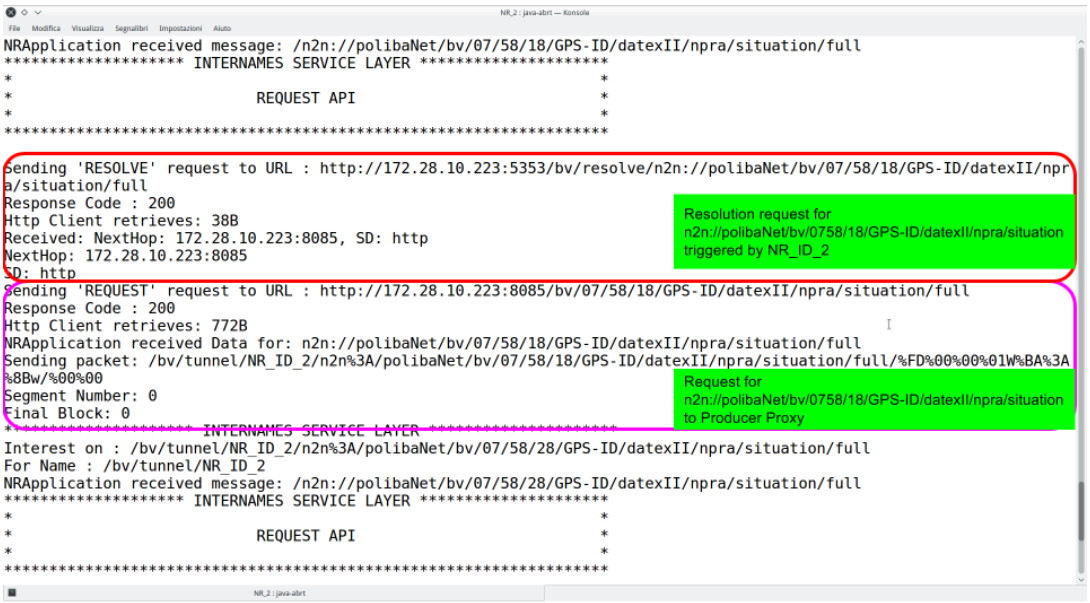

Figure 30: Logs reporting the set of operations executed by NR_ID_2 during the routing process for the first name.

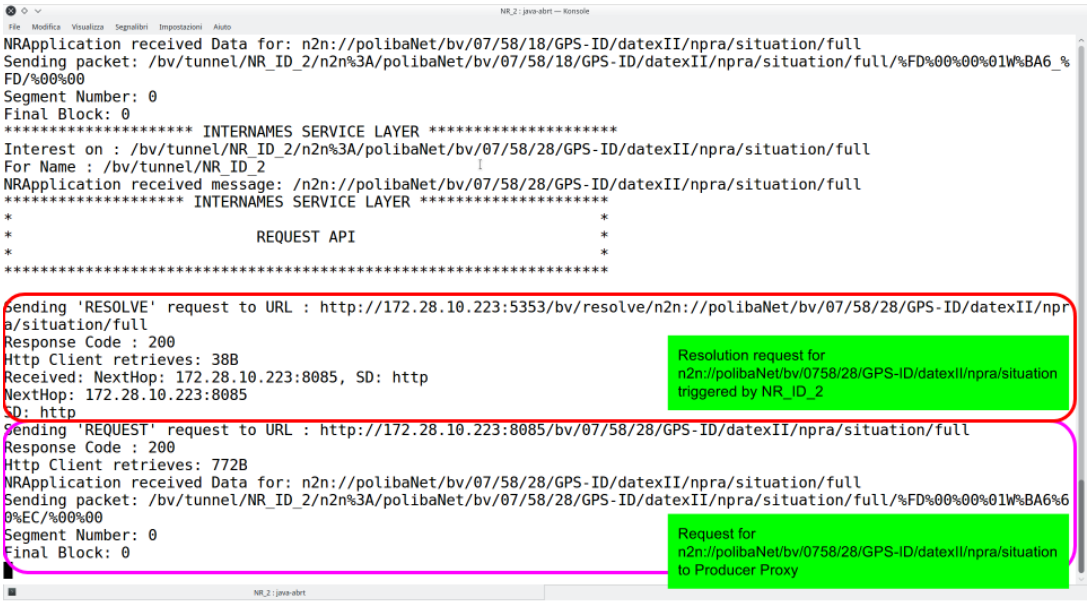

Figure 31: Logs reporting the set of operations executed by NR_ID_2 during the routing process for the second name.

back to the consumer.

- The **consumer** logs the reveived data (see Figure 32) and stores them within a dedicated folder (see Figure 33).

To conclude, the developed testbed demonstrated how the **consumer** is able to collect data generated in BONVOYAGE platform, without taking care about the details of the underlying technologies, the heterogeneous nature of the network, and the specific protocol architecture implemented in the network domains. The goal of the experimental testbed is indeed reached.

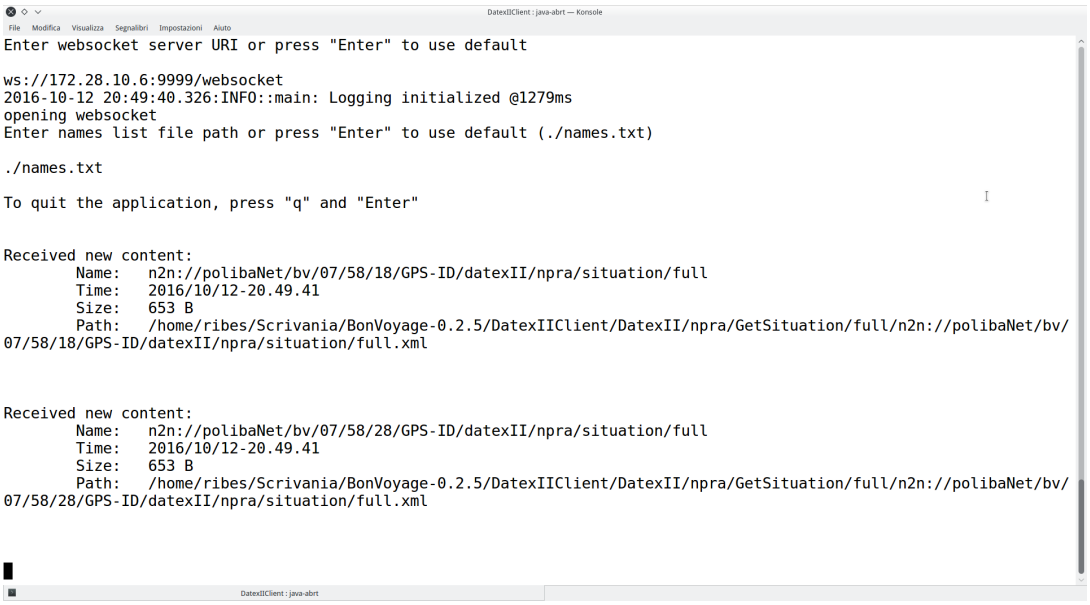

Figure 32: Logs of the consumer, generated when the requested contents are received.

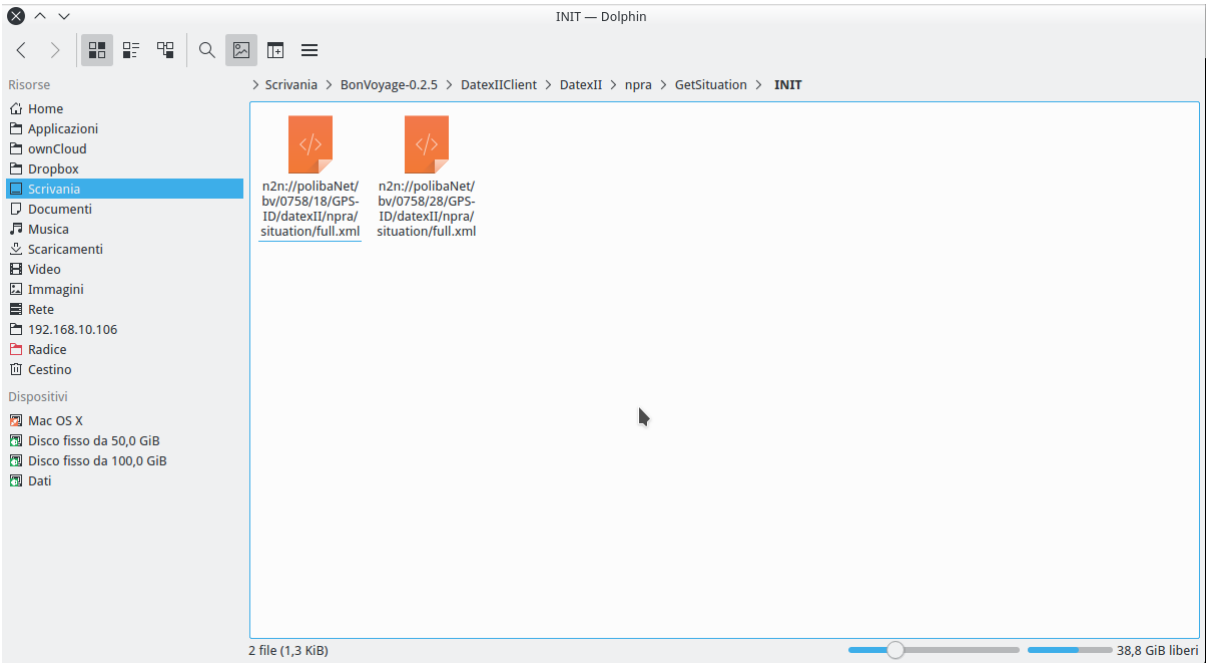

Figure 33: Files storing the contents requested by the consumer.

References

[1] "General Transit Feed Specification (GTFS)," 2016. [Online]. Available: https://developers.google.com/transit/gtfs/

[2] R. T. CEN Technical Committee 278, CEN/TC278 and T. Telematics), "Datex ii standard," 2016. [Online]. Available: http://www.itsstandards.eu/

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," *Proc. of ACM Int. Conf. on emerging Networking Experiments and Technologies (CONEXT)*, 2009.

[4] G. Piro, L. A. Grieco, G. Boggia, and P. Chatzimisio, "A survey of information-centric networking research," *Information-centric networking and multimedia services: present and future challenges*, 2013.

[5] M. Sadiku, S. Musa, and O. Momoh, "Cloud computing: Opportunities and challenges," *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, 2014.

[6] "FP7 COMET project," 2016. [Online]. Available: http://www.comet-project.org/

[7] "COMET Project. (2011, December) COMET deliverable 3.2: Final specification of mechanisms, protocols and algorithms for the content mediation system." 2016. [Online]. Available: http://www.cometproject.org/deliverables.html

[8] G. Piro, I. Cianci, L. A. Grieco, G. Boggia, and P. Camarda, "Information centric services in smart cities," *Journal of Systems and Softwares (JSS)*, 2013.

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

[10] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, , and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Surv. & Tuts.*, vol. 16, no. 2, pp. 1024–1049, 2014.

[11] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, , and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM*, pp. 181–192, 2007.

[12] "FP7 EU project, PURSUIT," 2016. [Online]. Available: http://www.fp7-pursuit.eu/

[13] "FP7 SAIL project," 2016. [Online]. Available: http://www.sail-project.eu/

[14] "FP7 CONVERGENCE project," 2016. [Online]. Available: http://www.ictconvergence.eu/

[15] "The Named Data Networking project," 2016. [Online]. Available: http://www.named-data.net/

[16] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, 2007.

[17] D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski, "Overlay networks and future of the internet," *Journal of Communications and Strategies*, p. 121, 2006.

[18] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

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

[20] A. Detti, N. B. Melazzi, M. Orru, R. Paolillo, and G. Rossi, "Opengeobase: Information centric networking meets spatial database applications," *arXiv:1607.00771v2*, 2016.

[21] D. T. et al., "Final updated architecture," *PURSUIT Publish Subscribe Internet Technology FP7-INFSO-ICT-257217*, 2013.

[22] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM*, pp. 181–192, 2007.

[23] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data," *ACM MobiArch*, 2010.

[24] D. Smetters and V. Jacobson, "Securing network content," *PARC, Tech. Rep. TR-2009-01*, 2009.

[25] D. Trossen and G. Parisis, "Designing and realizing an information centric internet," *IEEE Communications*, vol. 50, no. 7, pp. 60–67, 2012.

[26] J. Rajahalme, M. Sarela, K. Visala, and J. Riihijarvi, "On name-based inter-domain routing," *Computer Networks*, vol. 55, no. 4, pp. 975–986, 2011.

[27] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "On inter-domain name resolution for information-centric networks," *Proc. of the IFIP-TC6 Networking Conference*, 2012.

[28] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *IEEE Communications*, vol. 50, no. 7, pp. 52–58, 2012.

[29] N. Fotiou, K. Katsaros, G. Polyzos, M.Sarela, D. Trossen, , and G. Xylomenos, "Handling mobility in future publish-subscribe information-centric networks," *Telecommunication Systems*, 2015.

[30] D. Lagutin, "Redesigning internet-the packet level authentication architecture," *Licentiate Thesis, Helsinki University of Technology, Finland*, 2008.

[31] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 16, no. 7, pp. 422–426, 1970.

[32] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. C. de Blas, F. Ramon-Salguero, L. Liang, S. Spirou, A. Beben, , and E. Hadjioannou, "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 112–120, 2011.

[33] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," *Proc. of the IFIP-TC6 Networking Conference*, 2012.

[34] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," *ACM Workshop on Information Centric Networking (ICN)*, 2012.

[35] A. Detti, N. Blefari-Melazzi, S. Salsano, , and M. Pomposini, "Conet: A content centric inter-networking architecture," *ACM Workshop on Information-Centric Networking (ICN)*, 2011.

[36] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. BlefariMelazzi, "Transport-layer issues in information centric networks," *ACM Workshop on Information-Centric Networking (ICN)*, 2012.

[37] "NSF Mobility First project," 2016. [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[38] T. A. Baid and D. Raychaudhuri, "Comparing alternative approaches for networking of named objects in the future internet," *IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, 2012.

[39] T. Vu, A. Baid, Y. Zhang, T. Nguyen, J. Fukuyama, R. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 698–707, 2012.

Appendix: overview of concrete ICN architectures

At the time of this writing, ICN oriented projects include DONA [11], PURSUIT [12], SAIL [13], COMET [6], CONVERGENCE [14], and NDN [15]. Even if these architectures share the ICN communication paradigm (presented before), they differently address some other key aspects, including: naming schema and security, routing strategies, cache management, as described hereafter in more detail.

DONA

Data Oriented Network Architecture (DONA) [22] is one of the first complete ICN architectures. One of its most interesting aspect is that it radically changes Internet-like naming by replacing hierarchical URLs with flat names. Unlike URLs, bounded to specific locations via their DNS component, the flat names in DONA can be persistent even when information moves. Mobility is guaranteed and availability is preserved and improved as data can be cached and replicated at the network layer.

Each piece of information (or service) is associated with a principal and names consisting of the cryptographic hash of the principal's public key and a label. The principal is considered to be the owner of the corresponding information. The principals may name an entire web site or each individual web page within it. Names have several peculiar characteristics, as they are: flat, application and location-independent, and globally unique.

For the sake of clarity, practical Name Resolution and Data Routing cases are now going to be explained. Name resolution is provided by specialized servers called Resolution Handlers (RHs) which are distributed to be at least one for each Autonomous System (AS). RHs are interconnected and form an hierarchical name resolution service on top of the existing inter-domain routing relations. As a consequence, name resolution and data routing respect the established routing policies between ASs. Information objects are availabe since publishers send a REGISTER message with the name to the local RH. This creates a pointer stored to the principal. The RH is in charge of propagating this registration to the RHs in its parent and peering domains (as shown in Figure 34), following the established routing policies. Each intermediate RH has to store a mapping between the object's name and the address of the RH, forwarding the registration. This implies that registrations are replicated in RHs and, since all providers are peers with each other, RHs providers are aware of all registrations in the entire network. In details, Figure 34 shows paths 1 to 3, which indicate Registration messages. Moreover, paths from 4 to 7 are related with FIND messages. Paths from 8 to 11 are DATA messages.

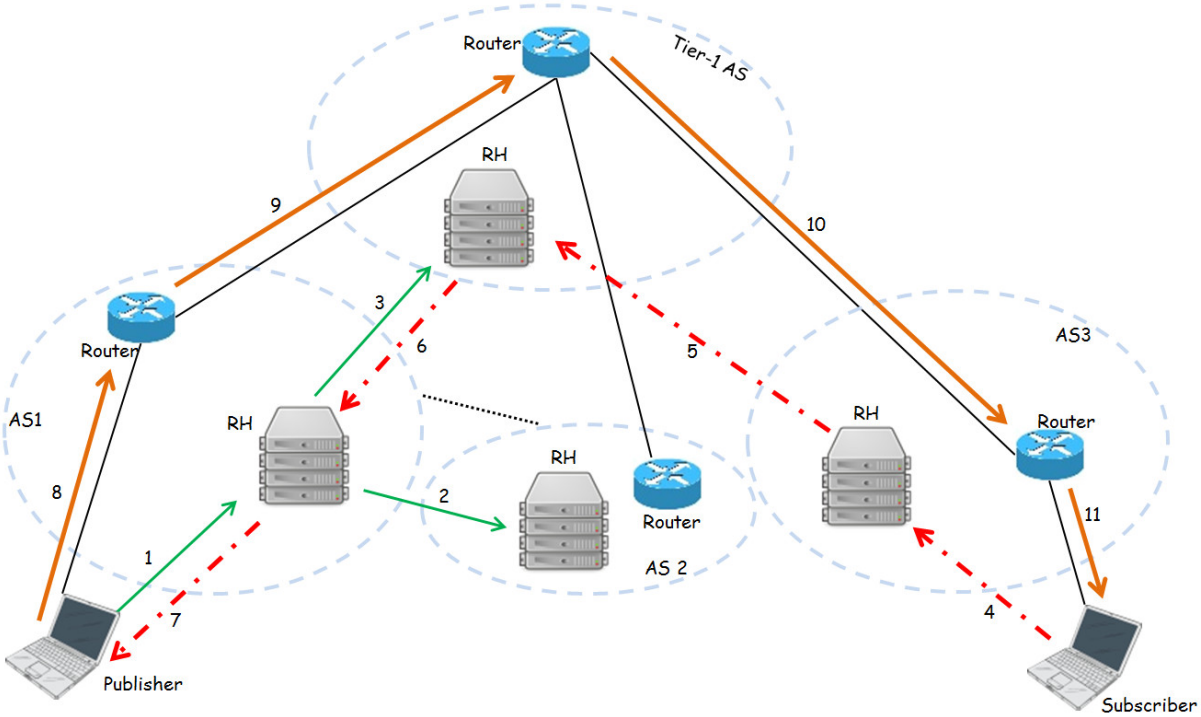

Figure 34: DONA architecture.

To locate an item, a subscriber sends a FIND message to its local RH, which has to propagate the message to its parent, according to its routing policies, until a matching registration entry is found. At that point, requests follow the pointers created by the registrations in order to reach the publisher; this process is guaranteed to succeed as long as the requested name exists. Data routing can either be decoupled or coupled with name resolution. In the decoupled case, as a FIND message reaches an appropriate publisher, data can be directly sent to the subscriber through regular IP routing and forwarding. Data transmission will then follow the routing policies for traffic between the publisher's and the subscriber's ASs.

This process requires global IP addresses and the architecture also offers a coupled option which relies on domain-local IP addresses only. In this case the FIND messages gather path-labels as they move from RH to RH, indicating the sequence of ASs crossed by the request. When the request reaches the publisher, these path-labels are used in the reverse order to retrace the path towards the subscriber. Even if the coupled option eliminates the need for global IP addresses and large BGP routing tables, since all path-labels have a local meaning, it enforces symmetric routing (at least, at the AS level) between requests and responses, even though this is not necessarily the case with regular BGP routing.

In DONA there are several additional features, such as:

- **Caching**: on-path caching is provided via the RH infrastructure. Before forwarding

the message, a RH that decides to cache a requested data object can replace the source IP address of an incoming FIND request with its own IP address. Any response will traverse the current RH and the returned data will be cached. Information may also be replicated off-path: in this case, when a RH receives multiple REGISTER messages for the same information, it will only maintain (and propagate upwards) the pointers to the best available copy. One of the parameter to classify copies can be the distance with a simple rule: the closest, the best.

- **Mobility**: mobile subscribers are allowed to issue new FIND messages from their current location and the RH infrastructure will provide them the best copy of the required content.

- **Security**: names in DONA are self-certifying, as they allow the subscriber to verify data through matching with the name requested. Data may either be mutable or immutable. In the former case, a client requesting an information object will receive both the public key of the principal as meta-data and a signature for the data object itself; this allows the data to be authenticated as coming from the specific principal. In the latter case, subscribers can simply verify that the label is the cryptographic hash of the information object.

NDN

The Named Data Networking (NDN) [15] aims to reshape the Internet protocol stack using various networking technologies below the waist for connectivity, including, but not limited to, IP. In NDN a strategy layer is used to mediate between the named data layer and the underlying network technologies, in order to optimize resource usage, selecting a link in a multi-homed node.

Names in NDN are hierarchical and may be similar to URLs, e.g. /aueb.gr/ai/main.html. However, NDN names are not necessarily URLs: their first part is not a DNS name or an IP address and they do not have to be human-readable. Each name component can be anything, including a dotted human-readable string or a hash value. This structure is considered a crucial aspect for scalability, as this allows name resolution and data routing information to be aggregated (if similar).

A request for a name is considered to match any piece of information whose name has the requested name as a prefix, e.g. aueb.graimain.html_v1_s1. This can be considered as a tree structure for addressing the first segment of the first version of the requested data. Subscribers can ask for the next data segment either directly requesting for the next information or asking for the next version by requesting the first sibling of aueb.graimain.html_v1.

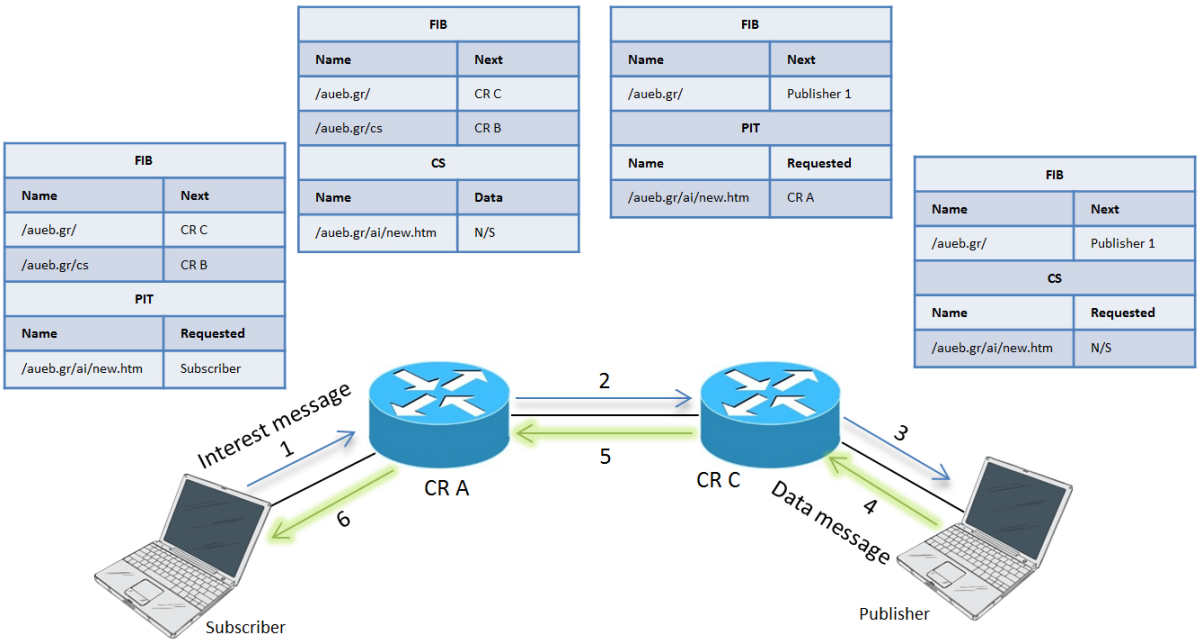

Figure 35: NDN architecture.

In NDN architecture, shown in Figure 35, data consumers issue INTEREST messages to request information objects (DATA packets), with the name of the requested/transferred information object. Messages are forwarded hop-by-hop by Content Routers (CRs); each CR maintains three data structures: the Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store (CS). FIB maps information names to the output interface(s) used to forward INTEREST messages towards appropriate data sources. PIT tracks the incoming interface(s) from which pending INTEREST messages have arrived. Lastly, the CS serves as a local cache for information objects that have passed through the CR. In Figure 35, paths from 1 to 3 are related to INTEREST messages whereas paths from 4 to 6 are DATA messages.

When an INTEREST arrives, the CR extracts the information name and looks for an information object inside its CS whose name matches the requested prefix. As long as something is found, it is immediately sent back through the incoming interface in a DATA packet, thus discarding the INTEREST. Otherwise, the router performs a longest prefix match on its FIB as to decide towards which direction this INTEREST should be forwarded. If an entry is found in the FIB, the router records the INTERESTs incoming interface in the PIT and pushes the INTEREST to the CR indicated by the FIB. In Figure 35 it is made clear that the subscriber sends an INTEREST for the name /aueb.gr/ai/new.htm (path 1-3). If the PIT already contains an entry for the exact name it means that the object has already been requested and the router adds the incoming interface to this PIT entry, discards the INTEREST and forms a multicast

tree for the information object.

When an information object matching the requested name is found at a publisher node or a CS, the INTEREST message is discarded and the information is returned in a DATA packet. This message is forwarded back to subscriber(s) in a hop-by-hop manner, based on the state maintained in the PIT(s). So when a CR receives a DATA packet it performs two tasks: it stores the corresponding information object in its CS and performs a longest-prefix match in its PIT to locate an entry matching the DATA packet. In case there are no matching entries in the PIT, the router discards the DATA packet as a duplicate.

In this architecture, name resolution and data routing are coupled. DATA packets follow the pointers left in the PITs by INTEREST messages so routing is symmetric. In order to populate the FIBs, NDN uses distributed routing protocols [3].

The followings are the most relevant NDN additional features:

- **Caching**: NDN natively supports on-path caching. Off-path caching is supported by delivering an INTEREST to any data source that may be hosting the requested information object.

- **Mobility**: When a subscriber moves in NDN, it can simply issue new INTEREST messages from its current location and the corresponding information objects will be delivered to its old location too. When it is the publisher moving, the FIBs pointing to it have to be updated, and this requires advertising again the name prefixes for the information it is hosting via the routing protocol. As this represents a very high overhead in high-mobility solutions, NDN uses the Listen First Broadcast Later (LFBL) protocol [23] to implement mobility in ad-hoc/opportunistic networks.

- **Security**: NDN supports the association of human-readable hierarchical information names with the corresponding information objects [24]. Each DATA packet contains a signature and the information so any node, including CRs, verify the binding between the name of the packet and the accompanying information.

PURSUIT

PURSUIT is an acronym standing for PURSUIT [12] and refers to an architecture that completely replaces the IP protocol stack with a publish-subscribe one. It consists of three different functions: rendezvous, topology management, and forwarding. When the rendezvous node matches a subscription to a publication, it asks the topology management function to create a communication route, used by the forwarding function to perform data transfer.

Information objects in PURSUIT are identified by a (statistically) unique pair of IDs, the scope ID and the rendezvous ID, the first grouping related information objects, the second indicating the actual identity for a particular piece of information [25]. Scopes are used to define sets of information objects within a given context as well as enforcing boundaries based on some dissemination strategy for the scope. If PURSUIT names are flat, scopes can be organized in scope graphs of variable forms, including hierarchies.

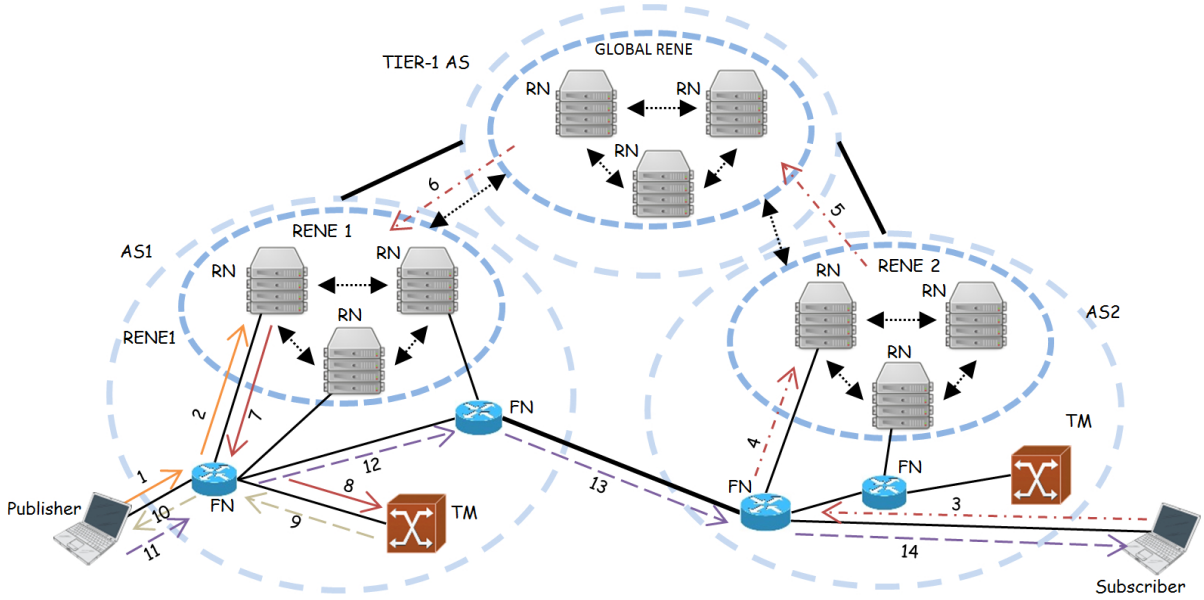

Figure 36: PURSUIT architecture.

Name resolution is handled by the rendezvous function, implemented as a group of Rendezvous Nodes (RNs), known as Rendezvous Network (RENE) [26], [27]. Name Resolution and Data Routing policies are graphically depicted in Figure 36. When a publisher wants to advertise an information object, it issues a PUBLISH message to its local RN which is routed by the DHT to the RN assigned with the corresponding scope ID. When a subscriber issues a SUBSCRIBE message for the same information object to its local RN, it is routed by the DHT to the same RN. A Topology Manager (TM) node will then create a route connecting the publisher with the subscriber.

The route is then sent to the publisher in a START PUBLISH message originating from the TM. After that, it is used to send the information object via a set of Forwarding Nodes (FNs). The TM nodes implement the topology management function executing a distributed routing protocol to discover the network topology.

Name resolution and data routing are decoupled in PURSUIT, since name resolution is performed by the RENE, while data routing is organized by the TMs and executed by the FNs. Name resolution may be time consuming, especially since DHT routing does not follow the

shortest paths between the communicating nodes. For the sake of clarity, Figure 36 explains working details highlighting paths 1 and 2, indicating PUBLISH messages. Paths from 3 to 6 are related to Subscribe messages. Paths from 7 to 10 are related to Delivery path request (more specifically, 7 and 8) and Star PUBLISH messages (paths 9 and 10). Paths from 11 to 14 are indicating DATA messages.

As it has been done in previous cases, here are identified:

- **Caching**: PURSUIT can support both on-path and off-path caching [28]. In the former case, forwarded packets are cached at FNs to serve subsequent requests. This feature could not be effective as a consequence of the decoupled nature of name resolution and data routing. In the latter, caches operate as publishers, advertising the available information to the RENE.

- **Mobility**: Mobility in PURSUIT is facilitated by the use of multicast and caching [28]. Different types of mobility cases are considered, based on local or global host movement and technology interchange. Local subscriber mobility can be handled via multicast and caching. Global subscriber mobility is handled modifying the forwarding function of the architecture [29]. Publisher mobility is harder, since the topology management function has to be notified of the publishers new position in the network.

- **Security**: encryption and individual packet signature are made possible thanks to the Packet Level Authentication (PLA) technique [30], with which packets are checked either at FNs or at their final destination, thus assuring data integrity and confidentiality. Flat names also permit self-certifying names.

SAIL

Scalable and Adaptive Internet Solutions (SAIL) [13], also referred to as **NetInf!** (**NetInf!**), includes many services, such as searching for information objects via keywords. SAIL architecture combines elements present in the NDN and PURSUIT approaches and can operate in a hybrid mode. Furthermore, it can be implemented over different routing and forwarding technologies, by introducing convergence layers to translate SAIL messages to actual network packets.

Information object names in SAIL provide some structure and can even be hierarchical, even though they are not caring about location or organizational information. SAIL defines the ni://A/L URI scheme in which names consist of an authority part A and a local part L. SAIL's names are considered flat for name comparison purposes. A subscription will only match

a publication if there is an exact name match between them. Routers can use longest prefix matching to determine how to route a message (so they can also be considered hierarchical when used for routing). This represents an intermediate and versatile approach combining PURSUIT and NDN.

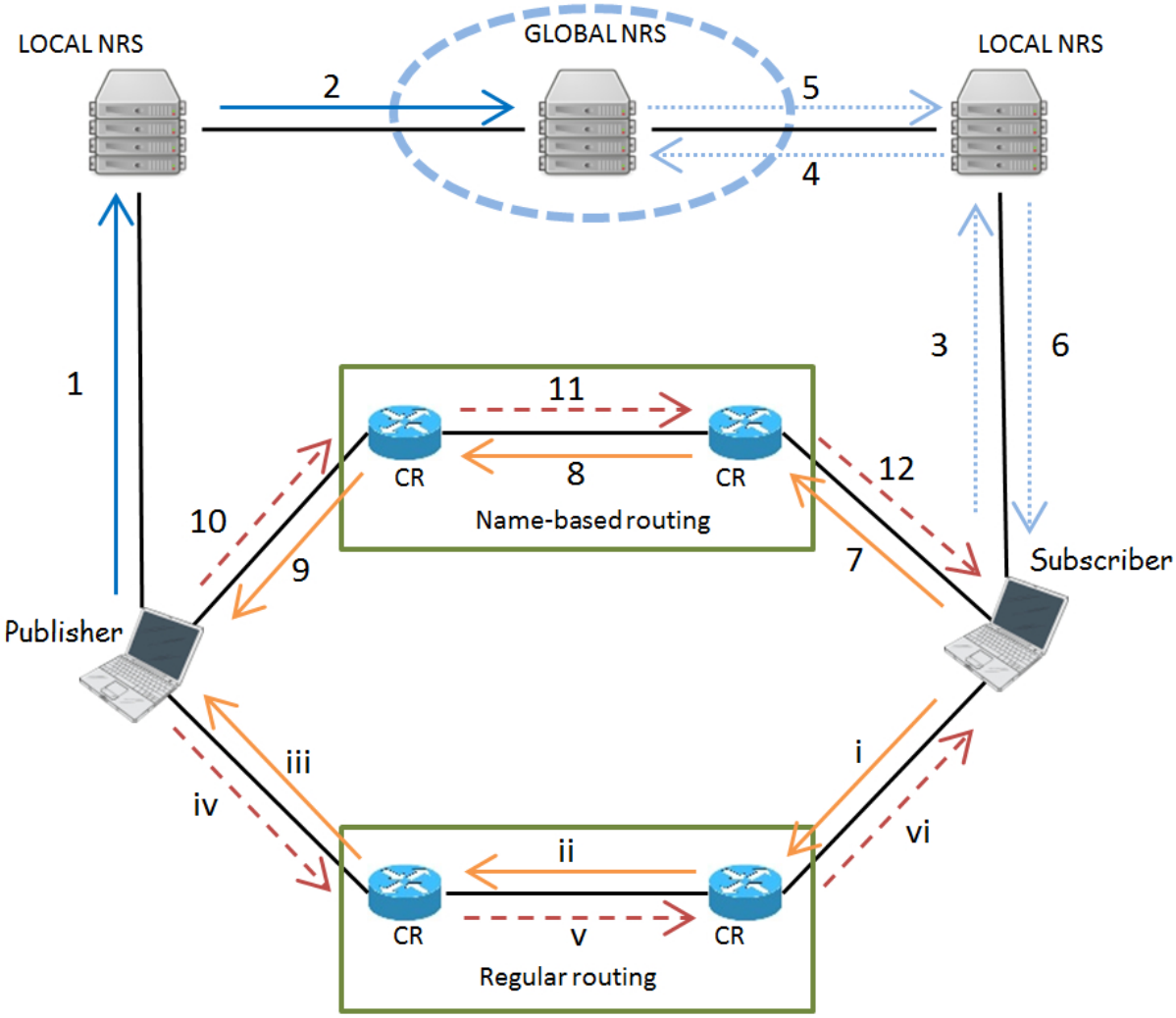

Figure 37: SAIL architecture.

In SAIL Name resolution and data routing can either be coupled or decoupled, moreover hybrid operations are possible, as shown in Figure 37, so it can be considered as a mix of hop-by-hop and partial paths.

In the decoupled case, a NRS is used to map object names to locators, so it's possible to reach the corresponding information object, and it works in a similar way to IP addresses. A publisher makes an information object available by sending a PUBLISH message with its locator to the local NRS that aggregates all the L parts [31] and sends a PUBLISH message to the global NRS (path 2). The global NRS stores the mapping between the authority A and the local NRS, replacing previous such mapping, if any. When a subscriber is interested in an

information object, it sends a GET message to its local NRS which consults the GLOBAL NRS (paths 3 and 4) to recognize a locator for the object (paths 4 and 5). The subscriber sends a GET message through the locator to the publisher, who responds with the information object in a DATA packet (paths iv - vi).

In the coupled case, a routing protocol is used to advertise object names and populate the routing tables of Content Routers (CRs). A subscriber sends a GET message to its local CR that propagates it towards the publisher. When the information object is found, it is returned via a DATA packet, reversing the exact path of the GET message. However, the GET messages accumulate routing directions along their path (7 9), which are reversed at the publisher or cached to reach the subscriber (10 12). For the sake of clarity, in Figure 37, paths 1 and 2 indicate PUBLISH messages. Paths from 3 to 6 are related to Request/Response resolution sequence messages. Moreover we have: paths from 7 to 9 (name-based routing GET paths), paths from 10 to 12 (name-based routing Data paths), paths from i to iii (for regular routing GET messages) and paths from iv to iv (for regular routing DATA messages).

In the hybrid mode, the NRS returns partial locators directing a GET message in one or more directions. A GET message can start with some routing hints from the NRS and then use name-based routing information stored in the CRs to reach its destination. A GET message can even start with the name-based routing information stored in the CRs and resort to the NRS for further routing, which might be useful when a CR does not have sufficient information to forward it.

SAIL's additional features are:

- **Caching**: together with on-path caching at the CRs, SAIL envisions the deployment of large scale information object caching and replication mechanisms in co-operation with the NRS. Caches in SAIL are hierarchical and part of a tree. Those higher up in the hierarchy have larger storage space in order to store popular objects, which otherwise would have been evicted by local caches due to their small size.

- **Mobility**: host mobility is supported having the NRS maintained topological information for each registered host. If a change of location happens, the moving host updates the topological information in the NRS where it is registered and a notification is sent to any nodes that are currently communicating with the mobile host.

- **Security**: SAIL architecture aims to cover name security, information integrity, authentication and confidentiality, and authorization and provenance through the inclusion of hash values in names. This makes self-certification of both the authority and the local

part possible.

COMET

The COntent Mediator architecture for content-aware nETworks (COMET) [6] optimizes information source selection and distribution. The core component of the COMET architecture is a Content Mediation Plane (CMP) that mediates between network providers and information servers, aware of both information and infrastructure. CMP is used in two different solutions: a coupled design called Content-Ubiquitous Resolution and Delivery Infrastructure for Next Generation Services (CURLING) [32] and a decoupled design that enhances information delivery without changing the underlying Internet [7]. COMET allows both subscribers and publishers to include location preferences for information explicitly.

The information names are provided by a Content Resolution System (CRS) when information is registered. This allows the naming system to associate similar contents and provides scaling possibilities exploiting existing relationships between information objects.

COMET supports two different Name Resolution and Data Routing approaches: coupled (Figure 38) and decoupled (Figure 39). In the coupled approach, as it is shown in Figure 38, a publisher sends a REGISTER message to its local CRS node. It issues a name for the information and stores the actual location of the information. Location is propagated upstream in the AS hierarchy using PUBLISH messages. Each parent CRS ends up with a pointer to its child CRS that sent the PUBLISH message (that could be limited to a specific area). A subscriber interested in some information issues a CONSUME message (path 3) to its local CRS. The message is propagated upwards in the CRS hierarchy until it reaches a CRS that has information about that name. When a match is found, the CONSUME message follows the pointers in the CRSs to reach the actual publisher. While the CONSUME message travels from the subscriber to the publisher, each CRS on the path provides forwarding state at the Content-aware Routers (CaRs) of each intermediate AS, back towards the subscriber. The publisher can send the corresponding data to the subscriber via these pointers.

The coupled approach in COMET looks similar to the DONA's name resolution and to NDN's data routing. Nevertheless, PUBLISH messages in COMET are not propagated to peering ASs but only to parents. This is useful to reach reduction of the state maintained at CRSs. As for data routing, while in NDN both name resolution and data routing use the same CRs, in COMET name resolution uses the CRSs while data routing uses the CaRs. This allows the CRSs in each AS to be more flexible when choosing the most appropriate paths between the available CaRs of that AS.

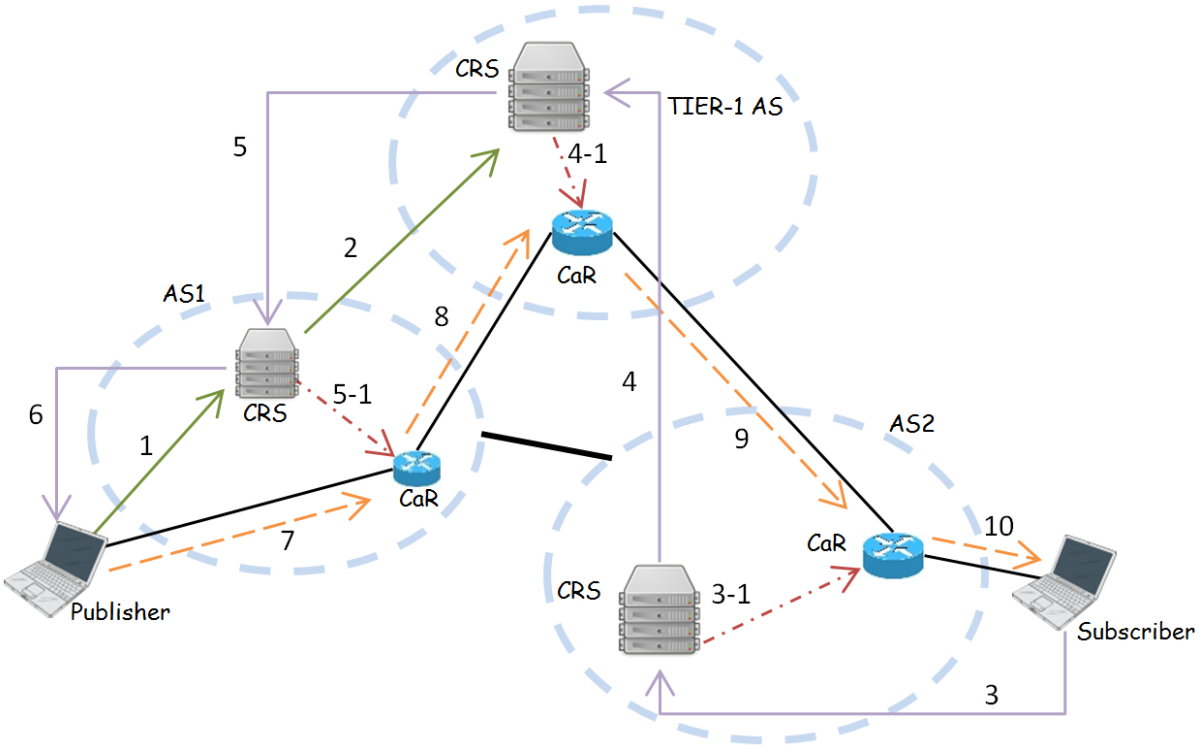

Figure 38: Coupled COMET architecture.

The decoupled approach in COMET Figure 39 provides for a CRS system which is similar to DNS as it splits the object namespace in a fixed, hierarchical manner. When a publisher wants to make some information available, it sends a REGISTER message to its local CRS. This is not propagated further because it must belong to the namespace assigned to that CRS. When a subscriber issues a CONSUME message, this is resolved by the root CRS to a pointer towards the publishers CRS. The subscribers CRS contacts the publishers CRS to get the location of the publisher. Then the subscribers Path Configurator (PC) contacts the publishers PC requesting a source route from the subscriber to the publisher. The source route is then returned to the subscriber. The reverse path is used by the publisher to return the information. In Figure 38, paths 1 and 2 are indicating REGISTER/PUBLISH paths. Paths from 3 to 6 are for Consume messages. Paths from 3-1 to 5-1 are for Route configuration. Paths from 7 to 10 are for DATA messages. In Figure 39, path 1 indicate REGISTER/PUBLISH message. Moreover, there are: path 2 (Consume message), paths from 3 to 6(Request/Response resolution sequence messages), paths 7 and 8 (Request/Response path messages), path 9 (Path information), paths from 10 to 12(Data Request) and paths 13 to 15(DATA messages).

Additional features in COMET are:

- **Caching**: COMET supports on-path and off-path caching. On-path caching is a product of name resolution, whereas off-path caching requires registering cached copies with the

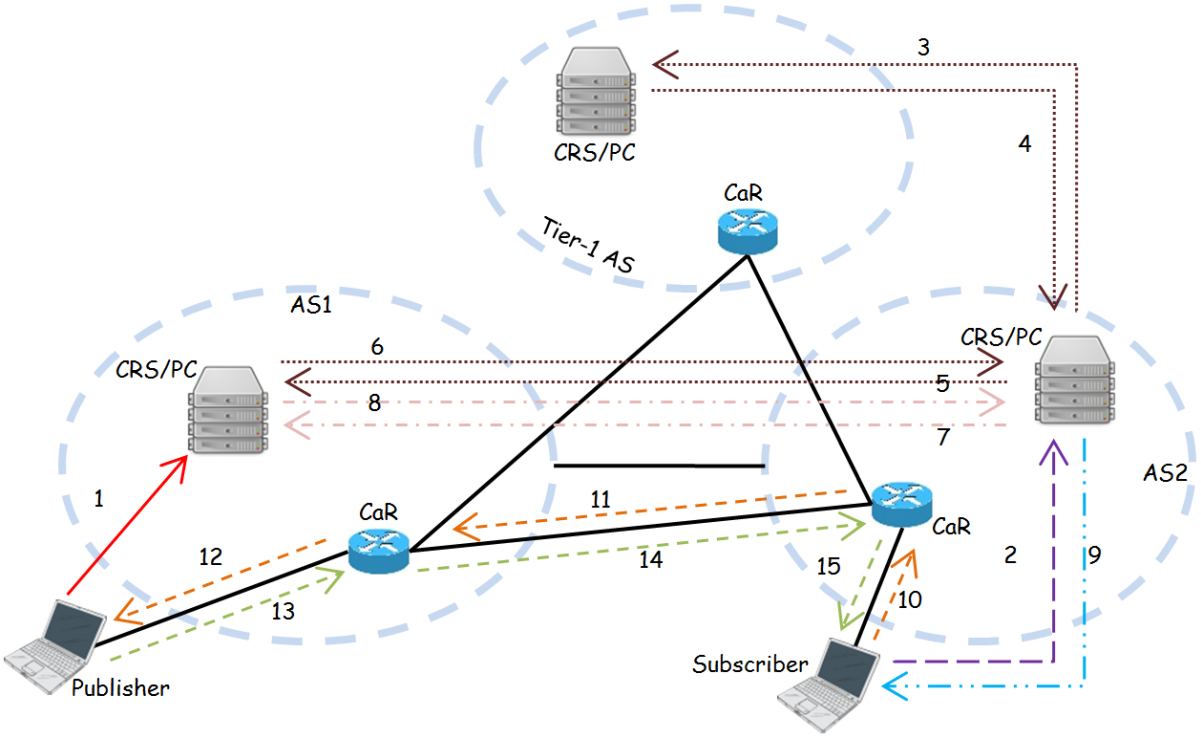

Figure 39: Decoupled COMET architecture.

CRS. COMET proposes two caching schemes: ProbCache and Centrality. The former is based on probability and it estimates the total traffic deriving this information from the requests it receives per unit time. The latter is based on capturing the number of times a specific node is contained on the shortest paths between all pairs of nodes in a network topology [33], [34].

- **Mobility**: COMET uses specialized mobility-aware CaRs placed at the edge of the access networks to support user mobility. They track users and information's mobility to predict future locations. If the subscriber moves to a different domain, new CaRs may need to be configured with routing state after the handoff.

- **Security**: COMET adopts security techniques from other ICN architectures [7]. Security provisioning in COMET may be simplified by the use of AS paths rather than global addresses in both the CRSs and the CaRs. This is supposed to prevent attackers from using arbitrary network paths to launch undetected attacks.

CONVERGENCE

The CONVERGENCE [14] project envisions an ICN-based Future Internet that facilitates user access to information, spanning from digital data and services to people and real-world objects.

Each object in CONVERGENCE is represented by a Versatile Digital Item (VDI), a common container for all kinds of digital information, based on the MPEG-21 specification. A COntent NETwork (CONET) [35] allows publishers to make available VDIs and subscribers to express interest in those VDIs.

CONVERGENCE's object names consist of a namespace ID and a name part, with formats determined by the namespace ID. While the default format of CONVERGENCE names is flat, hierarchical names or even URLs may also be used. The exact properties of the names depend on the specific namespace used.

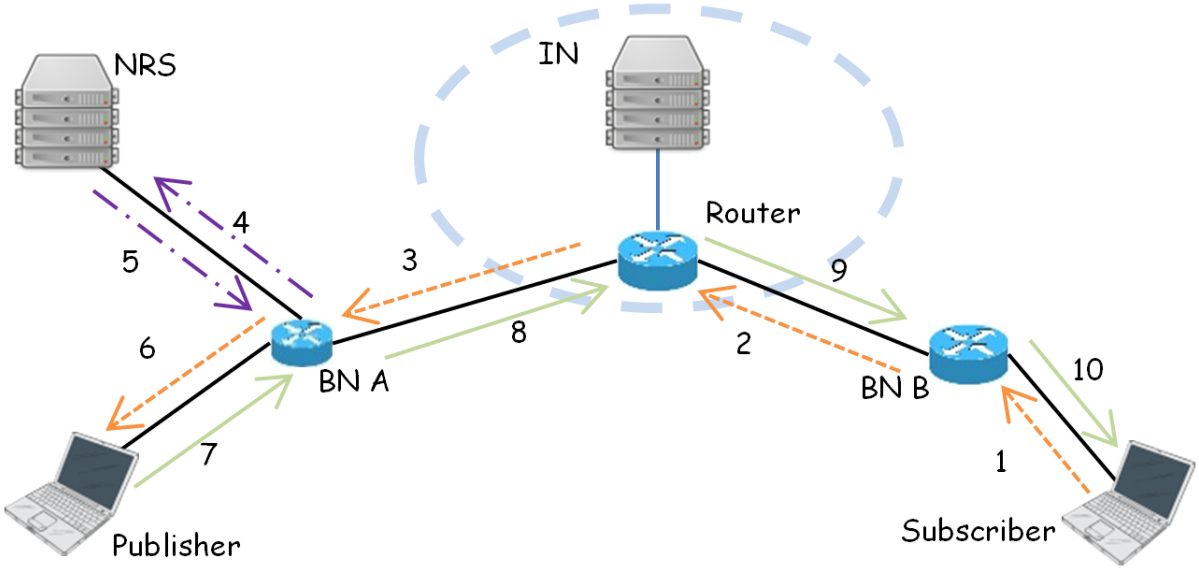

Figure 40: CONVERGENCE architecture.

The CONVERGENCE architecture, shown in Figure 40, has many similarities with NDN; indeed, its prototype has been implemented as a modification of the NDN prototype [36]. Subscribers issue INTEREST messages requesting an information object, which are forwarded hop-by-hop by Border Nodes (BNs) to publishers or Internal Nodes (INs) providing caching (paths 1-3 and 6). Publishers respond with DATA packets on the reverse path (paths 7-10). As to reduce the state requirements at the BNs, CONET differs from other approaches because BNs do not maintain name-based routing information for every advertised name prefix, but just for a part of them. Their routing table operates like a route cache and if there is no routing cache information for that name, and an INTEREST message cannot be forwarded, the BN consults an external NRS to find out how to forward the INTEREST (paths 4-5). The second difference is related to INTEREST messages, propagated as they accumulate the network addresses of the BNs they pass. This allows the publisher to route the DATA packet reversing this path information without maintenance of pointers at BNs. Thirdly, BNs do not

have to be directly connected; the path between two BNs can involve multiple hops. Unlike CRs in NDN, BNs map names to network addresses and not to interfaces. Name resolution and data routing are coupled. The paths taken by DATA packets are the reverse of those followed by the corresponding INTEREST message. The NRS is used if an appropriate route is not found at some BN. In Figure 40, there are: paths from 1 to 3 and path 6 indicating INTEREST, paths 4 and 5, indicating Name lookup and paths from 7 to 10, indicating DATA messages.

CONVERGENCE has some interesting additional features, such as:

- **Caching**: CONVERGENCE supports on-path caching as in NDN. Off-path caching and replication are supported by registering additional copies of an information object stored at IRNs to the NRS. Signaling overhead for this registration is not clear, as an NRS mechanism has not been implemented or defined.

- **Mobility**: CONVERGENCE supports subscriber mobility as they issue new INTEREST messages after a handoff. As for Publisher's mobility, it requires updating the NRS but the overhead is unknown, as noted above.

- **Security**: CONVERGENCE adopts the per DATA packet security approach, as it has been implemented in NDN. Due to the large overhead of meta-data required for signature verification, DATA packets are expected to be much larger than carrier packets. CONVERGENCE proposes performing security checks on information just at the DATA packet level at the subscriber, leaving BNs to only deal with the (smaller) carrier packets [36].

MobilityFirst

The MobilityFirst [37] project proposes an architecture treating mobile devices as first-class citizens [38]. It handles both mobility and wireless links, as well as multicast, multi-homing, in-network caching and security. The main principle is separation of names for all entities attached to the network. Each entity has a globally unique name, that can be translated into one or more network addresses at various points in the network. This allows messages to be dynamically redirected in order to follow a mobile device or content.

Figure 41 shows how communications are managed. Name Resolution and Data Routing are based on Globally Unique IDentifiers (GUIDs) associated to each network identity. They are translated to network addresses in one or more steps, via a Global Name Resolution Service (GNRS). A publisher asks the naming service for a GUID and then registers it with its network address in the GNRS (path 1). A GUID is mapped via hashing to a set of GNRS server

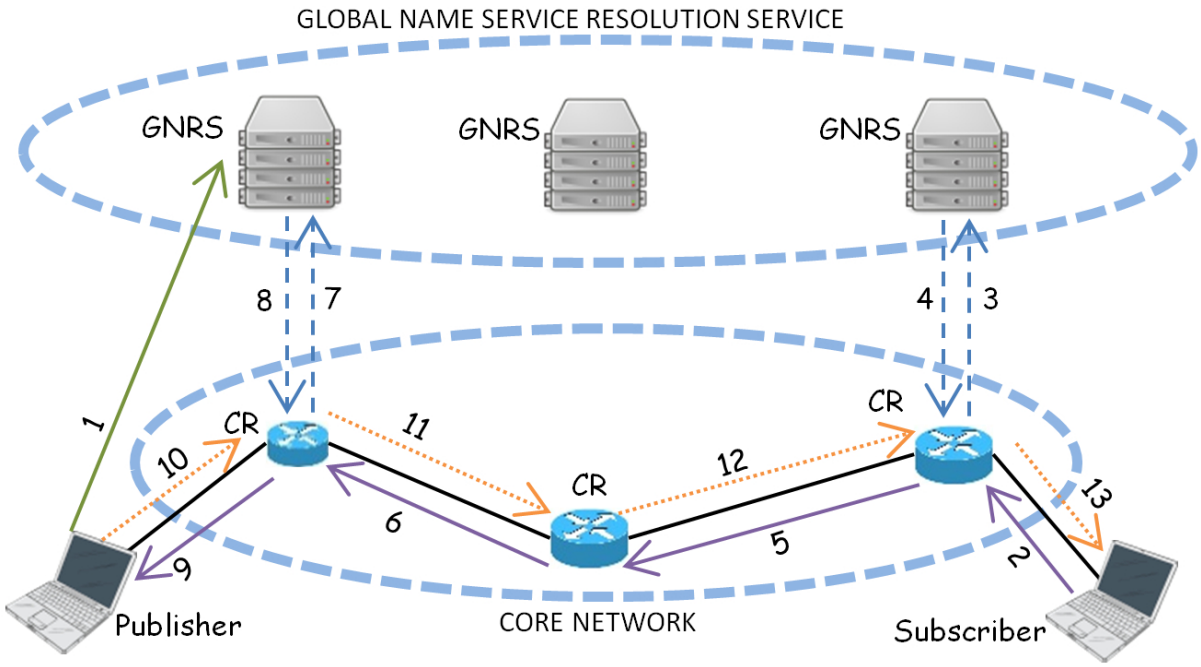

Figure 41: MobilityFirst architecture.

addresses, contacted using regular routing [39]. When a subscriber wants to receive some information, it sends a GET message that includes the GUID of the requested object, along with its own GUID for the response, to its local Content Router (CR) (path 2). CRs perform routing based on actual network addresses and ask the GNRS for a mapping the destination GUID with network addresses (path 3). The GNRS replies (path 4) with a set of network addresses, or with a source route and/or intermediate network addresses. CRs select one of these network addresses, add it to the GET message and forward it using a routing tables. The GET message includes both the destination GUID and the destination network address, and any CR along the path can consult the GNRS to receive an updated list of addresses for the destination GUID (paths 7-8). The publisher sends its response to the subscribers GUID, using the same procedure (paths 10-13). For instance, in Figure 41, path 1 indicate the REGISTER message, path 2, 5, 6 and 9 are for Request, paths 3, 4, 7 and 8 indicate Resolve GUID to locator and paths from 10 to 13 indicate the DATA messages.

For less dynamic services, MobilityFirst can translate each GUID to a network address once, as with DNS, and operate on network addresses only, ignoring the GUID. For more dynamic services, GUID may be translated multiple times as the first router asks the GNRS for the network addresses, which is connected with a given GUID and makes forwarding decisions. As a result, forwarding can be fast path, when the GNRS is bypassed, or slow path, when routers (re)consult the GNRS to obtain an updated list of network addresses.

Name resolution and data routing are decoupled as each message is delivered separately and

individually routed based on their destination GUIDs.

Even for MobilityFirst it is of great importance to describe:

- **Caching**: MobilityFirst supports on-path caching by opportunistically caching passing messages at intermediate CRs. This allows subsequent requests for the same GUID to be answered with the locally cached copy. Each time an information object is cached off-path or replicated, the GNRS is informed of the change and can update the corresponding GUID entry with the additional network addresses. While the GNRS can be repeatedly consulted as a message travels through the network, this is a slow-path operation, and each CR can implement its own policy on when to consult with the GNRS for additional cached copies.

- **Mobility**: As the aim is to address host, information, and entire network mobility, the first is handled by the GNRS. This unit must be updated when a network attached object changes its point of attachment. Network mobility is supported at lower levels and other distributed protocols can be used for disseminating routing updates.

- **Security**: trust model for name certification is decentralized. Independent naming organizations exist for mapping human-readable names to GUIDs. The GUID of an entity can be securely bound to that entity via cryptographic techniques.