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 D3.2:

Publish/Subscribe and security functionality

Deliverable Type:	R
Deliverable Number:	D3.2
Contractual Date of Delivery to the EU:	30.04.2017
Actual Date of Delivery to the EU:	29.04.2017
Title of Deliverable:	Publish/Subscribe and security functionality
Work package contributing to the Deliverable:	WP3
Dissemination Level:	PU
Editor:	Giuseppe Piro [CNIT]
Author(s):	Nicola Blefari [CNIT], Pietro Boccadoro [CNIT], Andrea Detti [CNIT], Luigi Alfredo Grieco [CNIT], Mauro Losciale [CNIT], Giuseppe Piro [CNIT], Giuseppe Ribezzo [CNIT], Giuseppe Tropea [CNIT]
Internal Reviewer(s):	Stephan Strodl [FLUIDTIME], Dag Kjenstad [SINTEF]
Abstract:	This deliverable presents the final architecture of the BONVOYAGE Communication System. Specifically, it explains all the networking aspects designed in WP3, thus including: (i) logical nodes and network architecture, (ii) high level functionalities and interaction mechanisms, (iii) front-end and networking APIs, (iv) publish-subscribe functionalities and (v) testbeds
Keyword List:	BONVOYAGE Communication System, networking, Information-Centric networking, Internames, Middleware, Namespace, Publish-Subscribe, API

Table of Contents

Abbreviations **8**

BONVOYAGE Glossary **9**

1 Introduction **12**

1.1 Deliverable rationale . . . 12

1.2 Quality review . . . 12

1.3 Executive summary . . . 13

1.3.1 Deliverable description . . . 13

1.3.2 Summary of results . . . 14

2 BONVOYAGE Communication System **16**

2.1 Design principles . . . 17

2.2 Involved entities . . . 19

2.2.1 Data Consumer . . . 20

2.2.2 Data Producer . . . 20

2.2.3 Consumer Proxy . . . 21

2.2.4 Producer Proxy . . . 21

2.2.5 Object Resolution Service . . . 21

2.2.6 Named Router . . . 22

2.2.7 Internames Rendezvous Node . . . 22

2.2.8 Name Resolution Service . . . 23

2.3 High level interaction among involved entities . . . 23

2.4 Naming . . . 26

2.4.1 Realm-based naming structure . . . 27

2.4.2 Geo-referenced names . . . 28

2.4.3 Hierarchical naming . . . 29

3 Front-end and Networking APIs **31**

3.1 Internames Service Layer and Networking APIs . . . 32

3.2 Front-end Application Programming Interfaces (APIs) and Data format . . . 39

4 Publish-Subscribe functionalities **42**

4.1 Request of a content . . . 44

4.2 Publication of a content . . . 45

4.3 Subscription to a content 46
4.4 Notification process 49
4.5 Request-Response communication scheme 50

5 Experimental testbeds **55**
5.1 Testbed focusing on networking APIs 55
5.2 Testbed leveraging the whole BONVOYAGE Communication System 65

Bibliography **78**

List of Figures

1 The BONVOYAGE Communication System . . . 17
2 Mapping functionality . . . 24
3 Publish functionality . . . 24
4 Subscribe functionality . . . 25
5 Notification functionality . . . 25
6 Routing operation . . . 26
7 Namespace structure . . . 27
8 Realm-based naming structure . . . 28
9 Geo-referenced structure . . . 29
10 Usage of APIs . . . 32
11 The Internames Service Layer and its APIs . . . 33
12 Technical details for the Internames Service Layer . . . 34
13 JSON messages structure . . . 41
14 Publish-Subscribe interactions . . . 42
15 Request-Response interactions - Items involved . . . 43
16 Publish-Subscribe interactions - Items involved . . . 43
17 Request of a content in BONVOYAGE . . . 44
18 Publication of content in BONVOYAGE . . . 45
19 Subscription to a content in BONVOYAGE . . . 47
20 Notification in BONVOYAGE . . . 49
21 Request-Response in BONVOYAGE . . . 51
22 ICN-IoT - Reference scenario implemented in the testbed . . . 56
23 Hardware components forming the developed testbed . . . 57
24 Sequence diagram. . . . 60
25 Steady state results for Humidity . . . 61
26 Steady state results for Brightness . . . 62
27 Steady state results for Temperature . . . 63
28 Steady state results for Acceleration . . . 64
29 Reference scenario implemented . . . 66
30 Logs reporting the triggering of front-end APIs by the Data Producer. . . . 67
31 Logs reporting the set of operations executed by the Producer Proxy during the publication process. . . . 68

32 Logs reporting the set of operations executed by the IRN logical node during the publication process. 69
33 Logs reporting the set of operations executed by the Consumer Proxy during the subscribe process. 70
34 Logs reporting the set of operations executed by the IRN logical node during the subscribe process. 70
35 Logs reporting the set of operations executed by the Consumer Proxy during the Request-Response process. 71
36 Logs reporting the set of operations executed by NRS_Net1 during the resolution process. 72
37 Logs reporting the set of operations executed by NR_ID_1 during the routing process 73
38 Logs reporting the set of operations executed by NRS_Net3 during the resolution process. 74
39 Logs reporting the set of operations executed by NRS_Net2 during the resolution process. 75
40 Logs reporting the set of operations executed by NR_ID_2 during the routing process 76
41 Logs of the Data Consumer, generated when the requested INIT and UPDATE contents are received. 76
42 INIT and UPDATE files stored by the Data Consumer 77

List of Tables

1 Abbreviations 8
2 BONVOYAGE Dictionary 11
3 Version Control Table 12
4 Involved entities in the BONVOYAGE Communication System 20
5 Functionalities in the BONVOYAGE Communication System 26
6 List of the APIs 32
7 Atomic networking operations 37
8 Messages generated in IP and NDN network realms. 38

Abbreviations

BONVOYAGE Abbreviations	
Abbreviation	**Definition**
6tisch	IPv6 over the TSCH mode of IEEE 802.15.4e
API	Application Programming Interface
App	Application
FIA	Future Internet Assembly
GPS	Global Positioning System
GTFS	General Transit Feed Specification
HTTP	HyperText Transfer Protocol
ICN	Information-Centric Networking
ICNRG	Information-Centric Networking Research Group
IEFT	Internet Engineering Task Force
IRTF	Internet Research Task Force
IoT	Internet of Things
IP	Internet Protocol
IRN	Internames Rendezvous Node
ISL	Internames Service Layer
ITS	Intelligent Transportation System
JSON	JavaScript Object Notation
NDN	Named Data Networking
NPRA	Norwegian Public Roads Administration
NRS	Name Resolution Service
OGB	OpenGeoBase
OM2M	Open Source implementation for M2M communication
ORS	Object Resolution Service
P2P	Peer-to-Peer
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
WPAN	Wireless Personal Area Network

Table 1: Abbreviations

BONVOYAGE Glossary

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

BONVOYAGE Glossary	
Term	**Definition**
6tisch	IPv6 over the TSCH mode of IEEE 802.15.4e IETF working group. It is in charge of releasing and maintaining a set of specifications that define the way that Time Slotted Channel Hopping mode of IEEE 802.15.4 standard can be integrated in embedded IPv6 networks based on the IPv6 over Low power WPAN
API	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.
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 content through request-response and publish-subscribe primitives.
Consumer Proxy	Logical node offering a standardized interface between Data Consumers and the rest of the BONVOYAGE Communication System.
Data Consumer	End user of the BONVOYAGE platform interested in getting 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. (e.g. NPRA)
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.

GTFS	Defines a common format for public transportation schedules and associated geographic information.
Internames	Customized instance of ICN, which identifies all elements involved in network communication, including contents, services, users, and devices by names. 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.
JSON	Syntax for lightweight data-interchange format.
Middleware	Software layer that hides the complexities and the differences of the underlying system's or hardware's technologies, exposing an abstraction layer to the overlying application and allowing the application developer to focus all his/her effort on the task at hand, without the distraction of orthogonal concerns at the system or hardware level.
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.
ONEM2M	Standard for Machine-to-Machine and IoT technologies.
OpenGeoBase	Discovery solution for travel-centric information.
Producer Proxy	Logical node offering a standardized interface between Data Producer and the rest of the BONVOYAGE Communication System.
SMARTM2M	Standard for Machine-to-Machine and IoT technologies.

WebSocket	Protocol enabling two-way communication between a client to a remote host.

Table 2: BONVOYAGE Dictionary

1 Introduction

1.1 Deliverable rationale

This Deliverable provides the final description of the **BONVOYAGE Communication System**. It extends the preliminary solution described in *Deliverable 3.1 - Networking* and introduces Publish-Subscribe functionalities. The work embraces the final outcomes of *Task 3.2 Publish/Subscribe and security functionality*.

The BONVOYAGE Communication System is a powerful communication bus used for disseminating travel-centric data and participatory sensing contents over an heterogeneous network architecture. 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.

Based on the work done in *Deliverable 3.1 - Networking*, this document investigates several aspects, including protocol interoperability, inter-domain routing, naming, and low-level communication primitives. Moreover, it explains the standardized Application Programming Interface (API) introduced to allow the communication process and describes the implemented versions of two different testbeds showing the main functionalities of the proposed solution.

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

As for security functionalities, all the envisioned solutions, will be described in *Deliverable 7.1 - Integration plan and guidelines*.

1.2 Quality review

The internal reviewers responsible of this deliverable are FLUIDTIME and SINTEF:

Version Control Table			
Version n.	**Purpose/Changes**	**Author**	**Date**
0.1	First draft	CNIT	14 April 2017
0.2	Quality Review	FLUIDTIME, SINTEF	27 April 2017
1.0	Final version	CNIT	28 April 2017

Table 3: Version Control Table

1.3 Executive summary

1.3.1 Deliverable description

The present Deliverable firstly provides a general definition of the main functionalities of the **BONVOYAGE Communication System**. The discussion focuses on "Internames", that is a customized Information-Centric Networking 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 main communication requirements envisaged in the BONVOYAGE project. Starting from these premises, Internames has been properly extended with Publish-Subscribe functionalities in order to fulfill additional requirements of the targeted Intelligent Transport System.

The big picture of the BONVOYAGE Communication System is presented in Section 2. Within it all relevant concepts, together with all the envisioned components and the design criteria, are described. Given the heterogeneous nature of the network architecture, becoming more and more demanding in terms of renewed connection and communication paradigms, Section 2 explains all the main components and the involved entities enabling message exchange. The namespace design criteria is discussed with a detailed explanation of its realm-based, geo-referenced and hierarchical structure.

In details, the system functionalities address several aspects, from network layer to application layer. Given the heterogeneous network architecture, in fact, data delivery is handled by means of specific communication protocols. In this scenario, Internames orchestrates name-to-name communications. 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. Named Routers and other involved logical nodes implement advanced networking functionalities to grant connection between clusters of nodes and effective data exchange. To better support a wide range of applications, the content exchange is handled through both request-response and publish-subscribe communication schema, thus exposing related primitives to upper layers. This is made possible by means of dedicated networking APIs. The BONVOYAGE Communication System foresees front-end APIs to ease the interaction between Data Consumer and Data Producer with two entities performing data aggregation, Consumer Proxy and Producer Proxy, respectively.

Section 3 provides a detailed description of the designed middleware solution, together with the envisioned APIs. In particular, the Internames Service Layer is deeply explained, with specific reference to its architecture, focusing the attention to the interactions between its main

components. To this end, the relationship between the networking APIs and the middleware solution is deeply investigated. Moreover, front-end APIs are described together with the data format used for messages exchange.

Functional details about the Publish-Subscribe communication scheme are presented in Section 4. The conceived approach is explained in detail, with specific focus on the networking operations performed by the APIs. In particular, the focus is about operations performed for requesting and publishing new (or updated versions of) content. The request-response communication scheme is deeply investigated in this Section as well.

With Section 5, concrete examples are discussed in order to demonstrate the proper behavior of the designed solution. In particular, the first testbed is an application of the ICN paradigm to the Internet of Things (IoT) domain. It implements a heterogeneous network, composed by two network realms: one based on the current internet architecture (for instance, IP realm), the other based on the ICN paradigm. These realms are connected each other through a Named Router. A single Data Producer belonging to the IoT domain has been attached to realm for modeling the current Internet. On the other hand, the Data Consumer is installed on a computer attached to the ICN realm. In the proposed example it is assumed that the consumer is interested in getting data, generated from the sensors available on top of one constrained device belonging to the IoT domain (e.g., environmental light). The logical nodes of the BONVOYAGE Communication System are implemented in a centralized fashion and installed on the named router. All the networking functionalities triggered by request-response and publish-subscribe primitives are technically described at the end of the section. In the second testbed, publish-subscribe functionalities have been developed. The aim of this experimental solution is to verify content generation mechanisms together with delivery functionalities to Data Consumers. In particular, Data Producer generates contents including specific information using a predefined JSON format (described in details in Section 3). A dedicated publication API directly manages such operation. Every time a content is generated, both initial and updates values are announced to the system.

An important difference when comparing the two experimental testbeds is that in the first one Proxies have not been implemented. In the first case, in fact, middleware functionalities are demonstrated with Data Producer and Data Consumer directly connected to Internames. These interactions only require networking APIs.

1.3.2 Summary of results

In summary, this deliverable provides the following results:

- the definition of the design principles of the BONVOYAGE Communication System;
- the description of all the logical nodes and entities involved in the communication, for instance Data Consumer, Data Producer, Consumer Proxy, Producer Proxy, Object Resolution Service, Named Router, Internames Rendezvous Node and Name Resolution Service;
- the description of the naming scheme;
- a detailed description of the front-end and networking APIs developed to support dissemination of travel-centric information;
- the implementation of Publish-Subscribe functionalities by means of a dedicated set of APIs;
- a presentation of the deployed testbeds proving the efficiency of the envisioned solution.

2 BONVOYAGE Communication System

The BONVOYAGE Communication System is a communication bus that targets the efficient and scalable dissemination of travel-centric contents across a heterogeneous network infrastructure. In order to efficiently support data exchange in this multi-domain environment, the BONVOYAGE Communication System is designed by properly extending a specific instance of Information-Centric Networking, namely "**Internames**" [1]. This is a novel network architecture, evolving from the ICN paradigm, that is designed to allow name-to-name communications in network systems made up by different realms (for instance, clusters of nodes that rely on heterogeneous networking primitives and protocols).

Internames is deployed on large scale as an overlay network. Within it, all elements involved in the network communication (including contents, services, users, and devices) are identified by names. In fact, a hierarchical name-space is designed to identify contents, users, Named Routers, and all the Internames logical nodes (also referred to as "Involved entities"), as well as any operations triggered at the network layer by the Internames APIs. The core network of Internames is based on a concrete ICN implementation, namely **Named-Data Networking (NDN)** [2][3].

Moreover, to support data dissemination in a way that is transparent from the underlying communication technology, and that simultaneously guarantees a standardized interaction among different applications, a powerful middleware layer, namely **Internames Service Layer (ISL)** is deployed. In particular, dedicated logical nodes are disseminated in the network in order to offer: (i) search engines, (ii) message routing across realms, and (iii) publish-subscribe functionalities. In fact, any device involved in the communication, like Data Consumers, Data Producers, Named Routers, and logical nodes, implement an instance of the ISL 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.

The development of such a complete solution is perfectly in line with the main objectives stated by the **European Directive 2010/40/EU**. It defines, in fact, a legal framework for developing specifications to make Intelligent Transport System interoperable across heterogeneous organizations [4]. It addresses functional and technical criteria to reach interoperable and seamless Intelligent Transport System (ITS) services. The shared adoption of guidelines and non-binding measures facilitate cooperation, thus favoring the realization of a cleaner, safer and more efficient transport system. Among priorities, there is the need to provide real-time traffic information, free of charge for users. In details, the main challenges are: (i) optimal use

of roads, (ii) real-time traffic monitoring and travel data management, (iii) continuity of traffic, (iv) freight management, and (v) road safety and security applications.

A big picture of the BONVOYAGE platform, and its communication system, is depicted in Figure 1. In particular, it describes the NDN core network working between several realms, each relying on IP suite technology, for data dissemination. To each realm, several consumers and producers can communicate taking advantage of WebSocket connections. A set of dedicated routers facilitate the inter-realm communications in this heterogeneous scenario. More technical details will be provided in the following Sections.

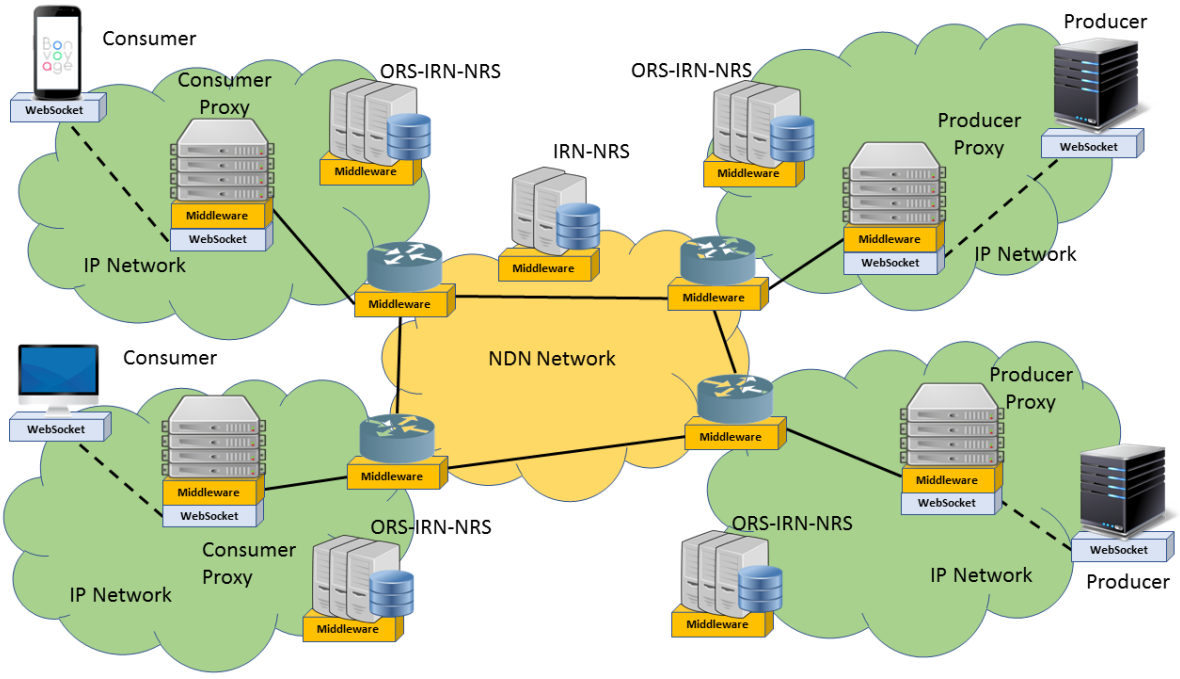

Figure 1: The BONVOYAGE Communication System

2.1 Design principles

The BONVOYAGE project, and in particular the BONVOYAGE Communication System, is aimed at creating a networking architecture that properly addresses issues and challenges due to both the extremely **heterogeneous** nature of ICN solutions and the interoperability with current Internet structure. As a matter of fact, only few portions of the network can be really re-designed from scratch, thus integrating pure ICN communication principles (clean-state solution [5]). In spite of the numerous ICN architectures proposed, many pieces of the current Internet still use the conventional Internet Protocol (IP). Nevertheless, to push the evolution of the Internet towards the information-centric approach [6], ICN could be quickly deployed as an overlay network, e.g., on top of the existing IP suite [7]. From one side, network nodes

can be grouped into clusters, simply referred to as realms. In realms, the data delivery is handled by means of common communication protocols. From another side, network realms can be connected one with the other through dedicated Named Routers, thus requiring advanced internet-working solutions.

The BONVOYAGE Communication System is designed to support some key features, here described in brief:

- The **design** of a scalable communication protocol based on the ICN principles. The main objective is to enable, and easily support, the exchange of travel-centric information across multiple domains. Moreover, particular attention is devoted to sensing contents across heterogeneous solutions. To this aim, the BONVOYAGE project builds upon on **Internames**. This instance of the ICN paradigm evolves from the baseline **host-to-name** principle to a **name-to-name** principle. Given this aspect, names are widely used to identify not only contents, but logical entities involved in communications and services. One of the main advantages when considering this approach is related to the fact that it does not require any statical bound to the physical location where the information is stored. Moreover, source/destination entities and services are included [1]. In addition, it natively allows: (i) the data exchange across heterogeneous network realms (which represents one of the main challenges of the BONVOYAGE project), and (ii) the Request-Response and the Publish-Subscribe mechanisms.

- The **development** of a scalable name-space through which uniquely identifying travel-centric information and sensing data. Names are also used to address Data Consumers, Data Producers, and any operation handled at both application and network layers.

- The **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. In this context, Request-Response and Publish-Subscribe communication schema are the main mechanisms through which applications may fetch contents from the network [8]. In detail, with the Request-Response scheme, data are retrieved in a synchronous way. Therefore, every time a Data Consumer (e.g., a journey planner application) fetches 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 Data Consumer (e.g., a journey planner application) issues a subscription request for a given content of interest. Then, every time

a new content (or even an updated version of an old one) is generated, the external source of information is in charge of delivering that data to all the subscribed Data Consumers or applications.

- The **implementation** of both networking and front-end APIs to enable communication among all the involved entities. The realization of such an objective has required the implementation of specific APIs aimed at enabling communications between entities involved in the envisioned network architecture. The APIs offered by Internames are used, for example, to announce new contents as long as to subscribe to them. Moreover, in order to ease the interaction between Data Consumer and Data Producer with the BONVOYAGE platform, dedicated front-end APIs have been designed. The development of such interfaces are used in order to allow, by means of dedicated proxies, both Data Consumers and Data Producers to connect to the platform without taking care about the underlying networking technologies.

In the rest of the present Deliverable, technical details will we provided in order to clarify the role of every single entity involved. Moreover, the interaction mechanisms through which data communication takes place will be discussed.

2.2 Involved entities

The aim of the present section is to describe each of entities involved in messages exchange, highlighting the main interaction mechanisms through functional details. In brief, Internames offers search engines, message routing across realms, and Publish-Subscribe functionalities through the following entities (shown in Figure 1): **Data Consumer**, **Data Producer**, **Consumer Proxy**, **Producer Proxy**, **Internames Rendezvous Node**, **Name Resolution Service** and **Named Router**. All of them are reported in Table 4. The aim of this table is to summarize which of the logical nodes belonging to the BONVOYAGE Communication System are inherited by the Internames implementation. As a consequence, it highlights those that are introduced thanks to the present contribution.

Logical Nodes	From Internames	New in BONVOYAGE
Data Consumer	✓	
Data Producer	✓	
Consumer Proxy		✓
Producer Proxy		✓
Object Resolution Service (ORS)	✓	
Internames Rendezvous Node (IRN)		✓
Name Resolution Service (NRS)	✓	

Table 4: Involved entities in the BONVOYAGE Communication System

2.2.1 Data Consumer

Data Consumers are the end users of the BONVOYAGE Communication System. They are connected to the network in order to retrieve travel-centric information for planning and optimizing transportation services, and their updates (if any). They connect to the BONVOYAGE platform as they are interested in collecting real-time data belonging to a set of names/topics. The list of names/topics is obtained through external search engines, for instance ORSs, described later on in the present Section. The Data Consumer uses a front-end API to send the aforementioned list to the Consumer Proxy. This API will be described in details in a dedicated Section of the present Deliverable.

2.2.2 Data Producer

Data Producers are the entities generating travel-centric contents and exposing them to the platform. For each travel-centric content, the Data Producer generates initial data (namely **INIT**) and punctual updates (namely **UPDATE**). Initial data are delivered to the Data Consumer as soon as it joins the platform and issues an interest for a specific content/name. Punctual updates, instead, include any single modification of the data made available by the Data Producer and are delivered to subscribers as long as they are generated. Initial data, and punctual updates, are announced by the Data Producer to the reference Producer Proxy by using front-end APIs which will be described later on in the present Deliverable.

2.2.3 Consumer Proxy

Consumer Proxy offers a standardized interfaces for Data Consumers in order to allow its communication to the rest of the BONVOYAGE platform. The Consumer Proxy, in fact, receives requests from the Data Consumer and retrieves all the data to which Data Consumers have performed the subscription. Thanks to this mechanism, aggregation is made possible. This kind of interaction is enabled by means of high-level APIs offered by the **Internames Service Layer** (see Section 3 for details). Consumer Proxy is also responsible for providing a message aggregation capability for the Request-Response interaction with the consumers. In fact, within the same realm, several Data Consumers are allowed to communicate to the Consumer Proxy to ask for travel-centric information. Consumer Proxies are in charge of collecting subscriptions and notifying the content availability to Data Consumers. The interactions of Consumer Proxy with subscribed consumers exploits the WebSocket technology and the JSON data notation for message exchange. The message structure details will be provided in the dedicated Section 4 of the present Deliverable.

2.2.4 Producer Proxy

In a similar way to what happens for Consumer Proxy, the **Producer Proxy** works as an interface between Data Producers and the core network. In particular, it is an entity collecting data coming from external servers (for instance, Data Producers) and exposing them according to the realm-based, geo-referenced, and hierarchical name-space. Producer Proxies are also in charge of managing the publication requests for a new or updated content announced by a Data Producer. For these reasons the Producer Proxy can be considered as a data source. Moreover, thanks to this mechanism, aggregation is made possible. The interactions between Producer Proxy and Data Producers relies on both the WebSocket technology and the JSON data notation for message exchange. The message structure details will be provided in the dedicated Section 4 of the present Deliverable.

2.2.5 Object Resolution Service

ORS offers mapping functionalities (details are provided later on in the present Section) and plays the role of a search engine. In general, a Data Consumer that is interested in retrieving a given content (or even more than one) is not necessarily already aware of 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 reporting the list of names that maps the aforementioned meta-data.

Since ORS implements functionalities that are very similar to those already provided by online search engine (i.e., Google and Yahoo) in today's Internet, no further implementation details will be investigated.

Similarly, OpenGeoBase (OGB) [9], is a discovery solution for travel-centric information realized within the BONVOYAGE project framework. It is a distributed spatial database system, able to store geo-referenced information. It can be defined as a search engine able to perform discovery operations over a specified area of interest defined by each user (for instance, Data Consumers). In order to address ICN functional features, OGB implements: (i) range-queries, (ii) routing-by-name dispatch, and (ii) data insertion. The latter, in fact, is made possible as ICN natively provides pull services. Moreover, such functionality is used to modify the ICN routing plane, if needed. These features favor scalable deployment and exploitation. All the functional details and design principles of OGB will be clarified and deeply characterized in *Deliverable 3.3. - Travel-centric and participatory sensing services.*

2.2.6 Named Router

Named Router is an entity involved in communications between realms. To be more specific, it is able to connect two or more different realms together. Its main aim is to perform routing operations. Two main aspects have to be recalled here: tunneling and caching. The former is an operation through which this entity provides packet forwarding across entities in a heterogeneous scenario. It is performed thanks to the path resolution discovery offered by the Name Resolution Service. On the other hand, the latter is used to optimize content delivery. In fact, when a content has to be delivered to a Data Consumer, it may be temporarily cached. This allows to satisfy future requests for the same content. It is worth noting that this characteristic can significantly lower the traffic load within the core network, also improving aggregation capabilities as long as it is implemented to serve Consumer Proxies. In order to grant cached resources management (e.g. invalidation), proper strategies will be described in Deliverable 3.3.

2.2.7 Internames Rendezvous Node

IRN manages Publish-Subscribe functionalities and exposes interfaces to the Consumer Proxy and the Producer Proxy at the network layer. In detail, it is responsible for:

- Processing the subscribe requests for Data Consumers. This operation is triggered by the Consumer Proxy for each name of interest in the list sent by the Data Consumers;
- Tracking the set of contents available in the BONVOYAGE platform;

- Processing the announce requests for Data Producers. This operation is triggered by the Producer Proxy every time a Data Producer generates and announces punctual updates;
- Notifying subscribed Data Consumers through the Consumer Proxy every time a new (or updated) content of interest is available. This operation is triggered in conjunction with the processing of the announce request for punctual updates.

2.2.8 Name Resolution Service

NRS handles routing operations within the heterogeneous network infrastructure. Given the distribution of data providers, it finds the best routing path to execute forwarding operations for the requests generated by Data Consumers. NRS is contacted by the Consumer Proxy, or by other routers along the path, every time a request has to be sent towards a new network realm. This routing functionality receives, as an input, the name of the content to be retrieved and returns, as an output, the next hop of the path and the related network protocol to use.

2.3 High level interaction among involved entities

The present Section is dedicated to a preliminary description of the interactions taking place in the BONVOYAGE Communication System. Involved entities are exchanging messages in order to request for, or subscribe to, contents. Some operations have to be performed in the dissemination procedure, for instance to publish and announce the availability of a new content to logical nodes within the platform. In particular, here are described: (i) mapping and (ii) routing functionalities, but also (iii) publish, (iv) subscribe, and (v) notification services.

In Figure 2, the mapping functionality is depicted. In details, the Data Consumer wants to retrieve a content but it is not able to give any detail about the name. In order to complete this discovery operation, the Data Consumer sends a message with meta-data to the Object Resolution Service. The information contained within the message are related to the content of interest. The answer that the Data Consumer will receive contains the list of all the names matching the communicated indications. With this list, the Data Consumer is able to use the front end APIs to retrieve the matching names. The list of names can also be useful for performing subscriptions.

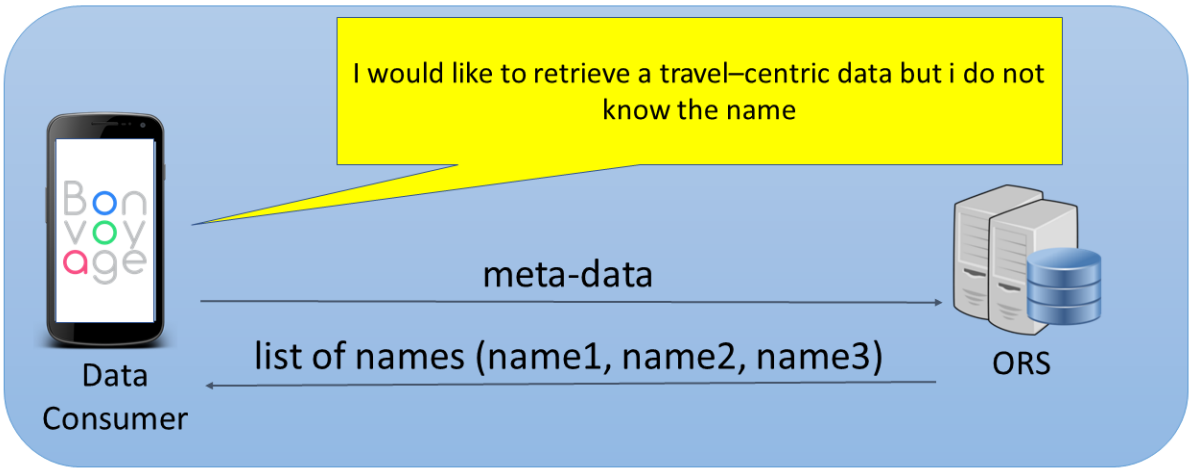

Figure 2: Mapping functionality

Figures 3 and 4 are reporting the high-level node interactions in the provisioned Publish-Subscribe schema. In the Publish interaction, the involved entities are Data Producer, Producer Proxy and IRN. In details, Data Producers interacts with the Producer Proxy to publish a new or update version of travel-centric contents, thus the Producer Proxy forwards the publish requests to the IRN.

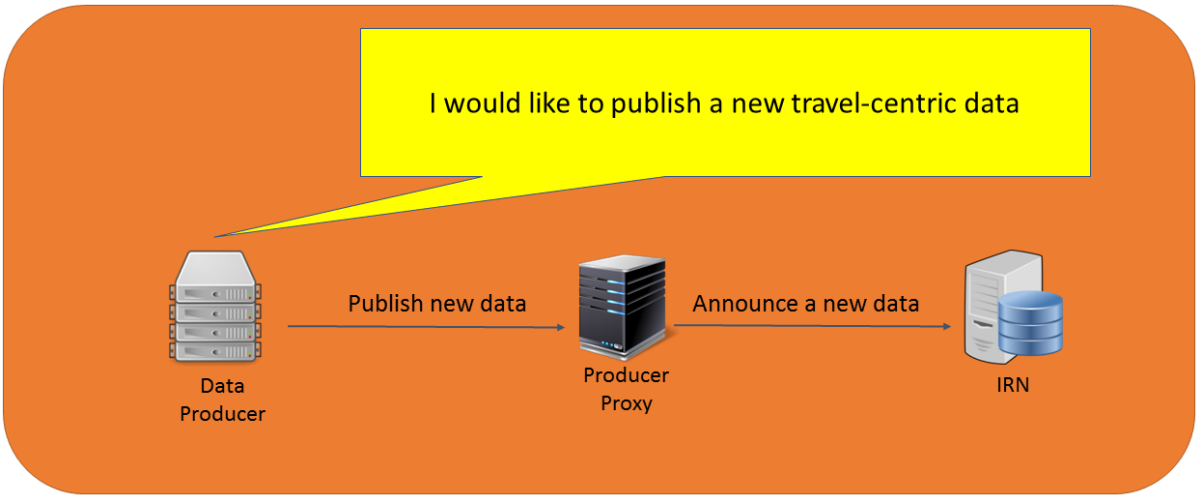

Figure 3: Publish functionality

On the other hand, for the Subscribe service the Data Consumers interact with the Consumer Proxy to ask and retrieve the name/content of interest for both initial data and future punctual updates (if any). It is really important to remark that several subscribed consumers interact directly with a single Consumer Proxy. Thanks to this, aggregation can be efficiently achieved.

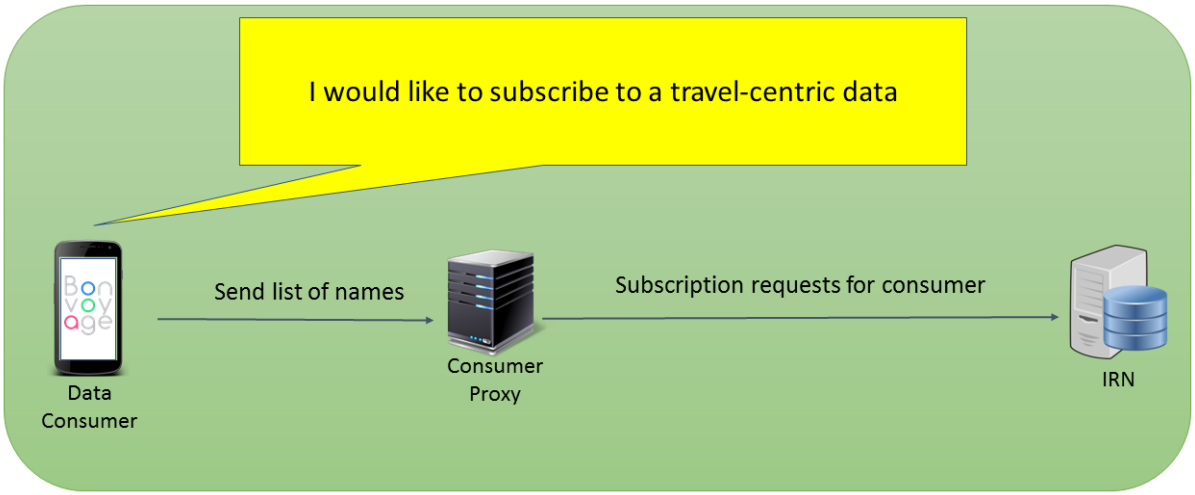

Figure 4: Subscribe functionality

The IRN logical node is involved in order to trigger the notification process when a new version of data is available in the platform. This process is shown in Figure 5. Moreover, the Consumer Proxy manages subscribe requests and notifications for connected Data Consumers.

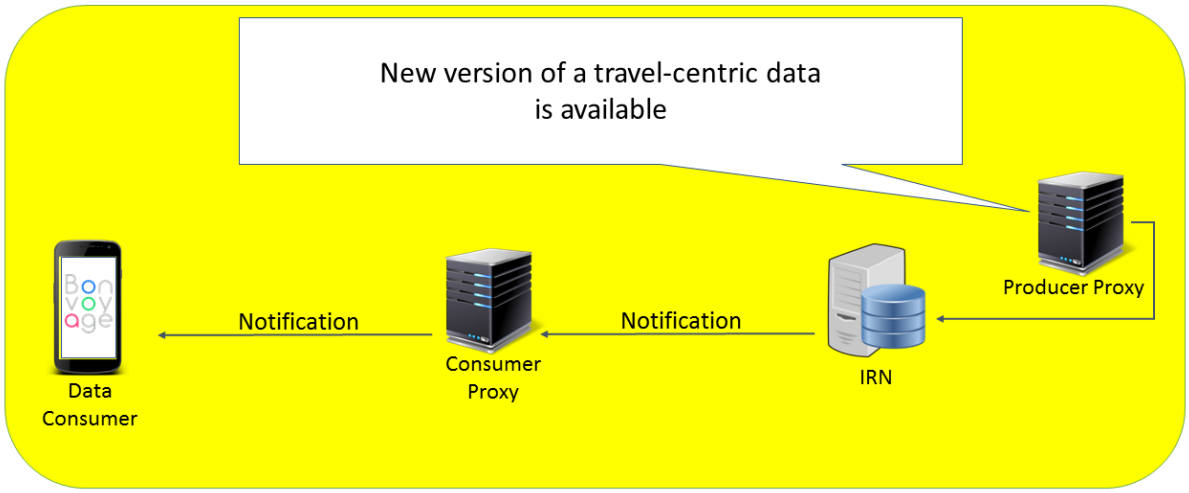

Figure 5: Notification functionality

In Figure 6 the routing functionality is explained. In detail, the interaction between Consumer Proxy and the Name Resolution Service consists of a message exchange in which the Consumer Proxy asks information about the correct path to which tunneling requests for travel-centric data. The answer, for this name, contains the details about the following hop, with a specific indication about the kind of realm in which the request is going to be forwarded. It is worth noting that the same operation is performed through the whole network until reaching the one in which the content is located. Routing operations are involved in Request-Response, discussed in details in Section 3 of the present Deliverable.

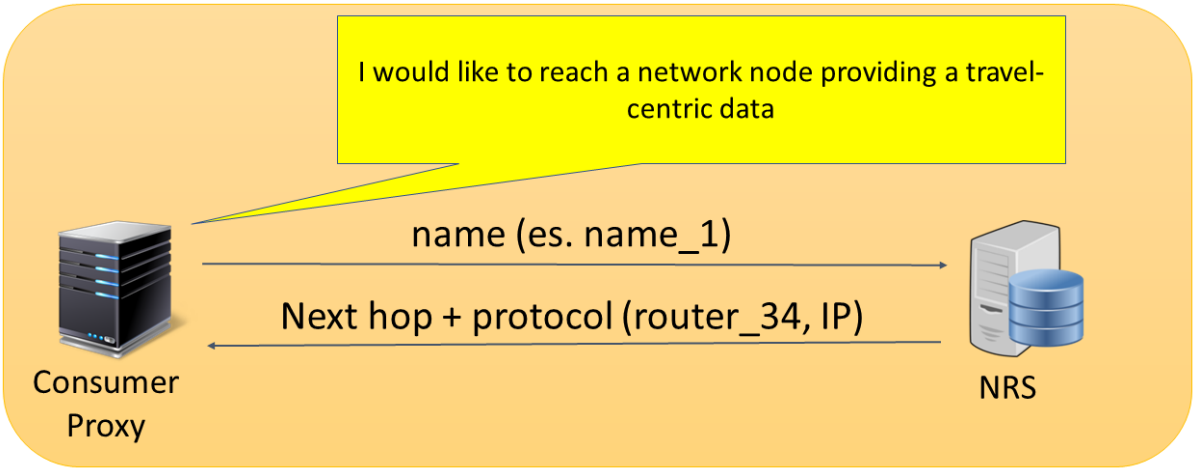

Figure 6: Routing operation

In Table 5 it is highlighted which of the functionalities implemented within the BONVOYAGE Communication Systems are inherited from Internames and which are innovations of this implementation.

Functionality	From Internames	New in BONVOYAGE
mapping	✓	
publication		✓
subscription		✓
request-response	✓	
notification		✓
routing	✓	

Table 5: Functionalities in the BONVOYAGE Communication System

2.4 Naming

In ICN, names are important because they are unique identifiers of data within the network. Until now, the rational of names creation has been taken for granted; same considerations have to be done about the kind of information they are addressing. The aim of this subsection is to clarify how they are conceived. According to ICN principles, the BONVOYAGE platform adopts a **realm-based**, **geo-referenced**, and **hierarchical** naming structure. It is useful to describe the availability of contents, devices, and services in a heterogeneous communication

infrastructure, based on the geographical area they belong to. The resulting name structure is reported in Figure 7.

Figure 7: Namespace structure

2.4.1 Realm-based naming structure

As said, the BONVOYAGE platform adopts a realm-based naming structure, which is useful to describe the availability of contents, devices, and services in a heterogeneous communication infrastructure. An example of this structure is:

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

where *[realm ID]* is the network realm where the content is available, and *[NAME]* is its unique name. This is the field aimed at identifying travel-centric and/or sensing data, as well as control messages exchanged in the BONVOYAGE Communication System during the provisioning of request-response and publish-subscribe services.

In Figure 8 a graphical description of the concepts explained is given.

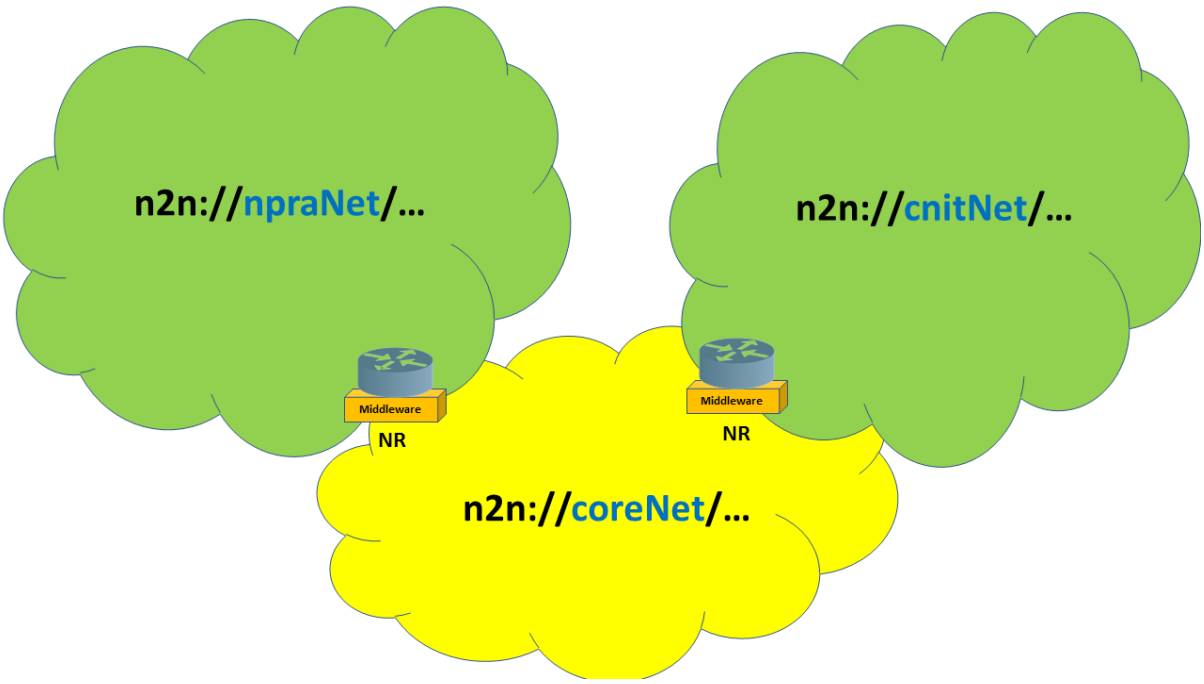

Figure 8: Realm-based naming structure

2.4.2 Geo-referenced names

The OpenGeoBase [9] approach to data mapping and retrieving implies structured access to information based on geo-referenced positions. A widely accepted way of indexing the spatial distribution of items on Earth is the usage of Global Positioning System (GPS) coordinates. Starting from this premise, a layered grid, made of different refined levels, allows precise items localization. In the OpenGeoBase approach, in fact, a number of grid regions, aligned with world's parallels and meridians, is used. Grid regions are called *tiles*; they are different in size and every size implies precision refinement. Moreover, tiles are aligned one with the other as they extend themselves starting from the same point. For the sake of clarity, 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 can be used to include the first decimal of the two, respectively. As the tiles grows in value (e.g. n = 1 or 2), value are considered up to the n-th decimal, e.g. level-2 tile is 8.43, 61.56. In this case, the selected region contains all points whose latitude and longitude start with 8.43 and 61.56, respectively. A level-n tile of the grid contains all those points having the same latitude and longitude but in a smaller region.

The structure of a name that contains all the numbers to the n-th level is:

$$[\mathbf{NAME}] = \mathbf{/bv/lng(0)/lat(0)/lng(1)lat(1)/.../lng(n)lat(n)/GPS\text{-}ID/[other\_info]}$$

Without loss of generality, it is assumed that the tile level used in the BONVOYAGE Communication System is level-1. The tile size in use covers a geographical area of 100 km^2 (10 km x 10 km). For instance, a content available in a geographical area having a tile described with GPS coordinates 8.4 and 61.5, is identified with the name:

$$[\mathbf{NAME}] = \mathbf{/bv/61/8/54/GPS\text{-}ID/[other\_info]}$$

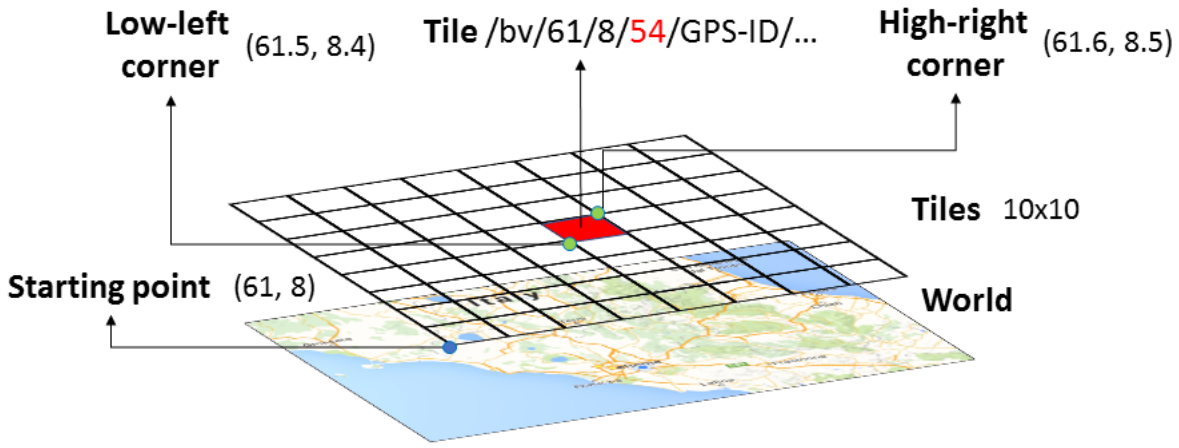

Figure 9: Geo-referenced structure

2.4.3 Hierarchical naming

In hierarchical namespaces, the names are composed by a series of strings, optionally encrypted, in which every name prefix identifies a sub-tree in the namespace. This allows 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 using an hierarchical tree structure. The BONVOYAGE platform adopts a hierarchical structure for the *[other_info]* part of the namespace, which specifies four key characteristics:

- the standard which the name refers to,
- the provider of the named content,
- the specific requested service,

- additional information (optional).

The resulting name is structured as follows:

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

For example,if the DATEX II standard [10] is used to provide information about Norvegian Public Roads Administration (NPRA) for a situation service, the name appears in the form:

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

3 Front-end and Networking APIs

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 involved entities, the provisioning of the high-level functionalities described so far is a complex task to accomplish. To this end, the **Internames Service Layer** has been designed to offer all of these features to the application layer by means of a standardized interface.

To better understand all of the concepts used from now until the end of the present Deliverable, it is important to focus our attention on two entities: **middleware** and **API**.

Middleware is the name given to a software solution providing services to software applications enriching those made available by 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 create the specific application. A middleware is aimed at connecting software components together and lies between the operating system and the applications on each side of a distributed computer network. As a design solution, it includes: (i) Web servers, (ii) application servers, (iii) 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 defined as a set of routines, protocols, and tools for building software and applications. A software component is defined in terms of its operations, inputs, outputs, and underlying types and functionalities. These characteristics have to be considered as independent from their respective implementations. The aim of an API is to design a program providing all the main building blocks. Using an API makes it possible for different solutions to work together, as part of the operative requirements. Same considerations can be done for technological background on which the different kind of networks are based. API development allows definitions and implementations without compromising the interface neither implying a new design process. An API can be used for a web-based system, an operating system or a database system. An API can also be used to develop Graphical User Interface components, that can be the end-user interfaces for both Data Producers and Data Consumer. In particular, such applications can be used when working on mobile devices.

With Figure 10 it is clarified where the different APIs are used. In particular, networking APIs are involved in communication within the core network, for instance between the proxies. Front-end APIs, instead, are used to ease the interaction between both Data Consumer and

Data Producer with the BONVOYAGE Communication System.

Table 6 summarizes the whole set of APIs made available to allow efficient data dissemination.

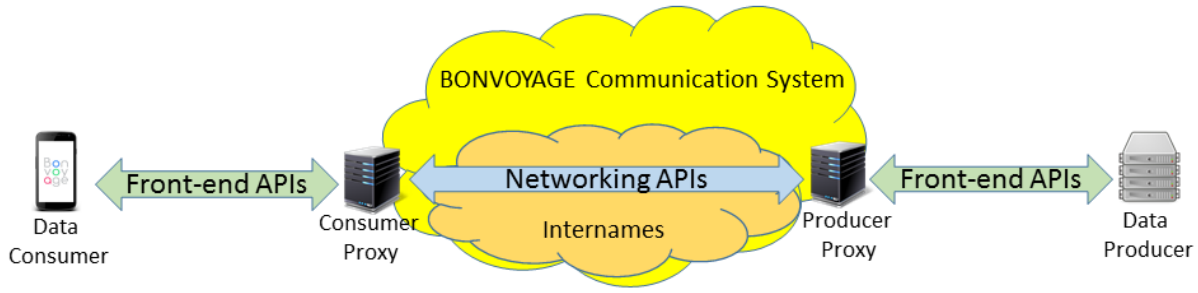

Figure 10: Usage of APIs

Front-end APIs	Consumer_Request
	Consumer_Subscribe
	Producer_Publish_Init
	Producer_Publish_Update
Networking APIs	Netw_Request
	Netw_Announce
	Netw_Subscribe
	Netw_Notify
	Netw_Del_Announce
	Netw_Del_Subscribe

Table 6: List of the APIs

3.1 Internames Service Layer and Networking APIs

The Internames Service Layer acts as a middleware between upper layer applications and all underlying communication technologies. As described in Figure 11, that middleware is distributed among all the actors of the BONVOYAGE platform (i.e., Data Consumers, Data Producers, Named Routers, and, in general, all the involved entities of the 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 underlaying communication technology.

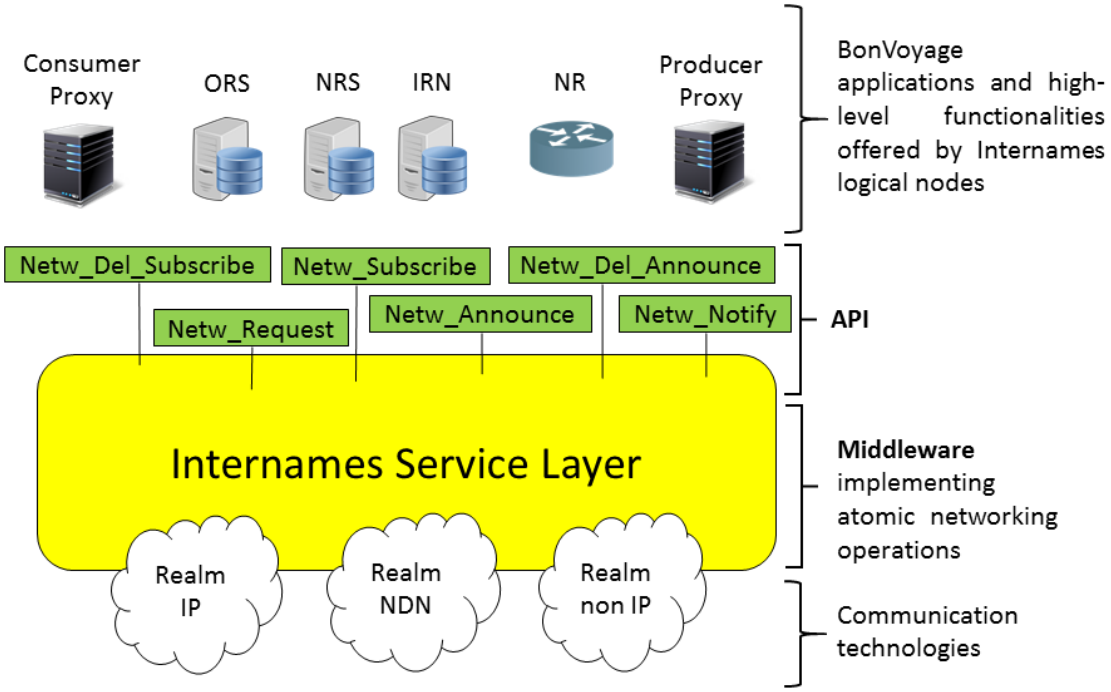

Figure 11: The Internames Service Layer and its APIs

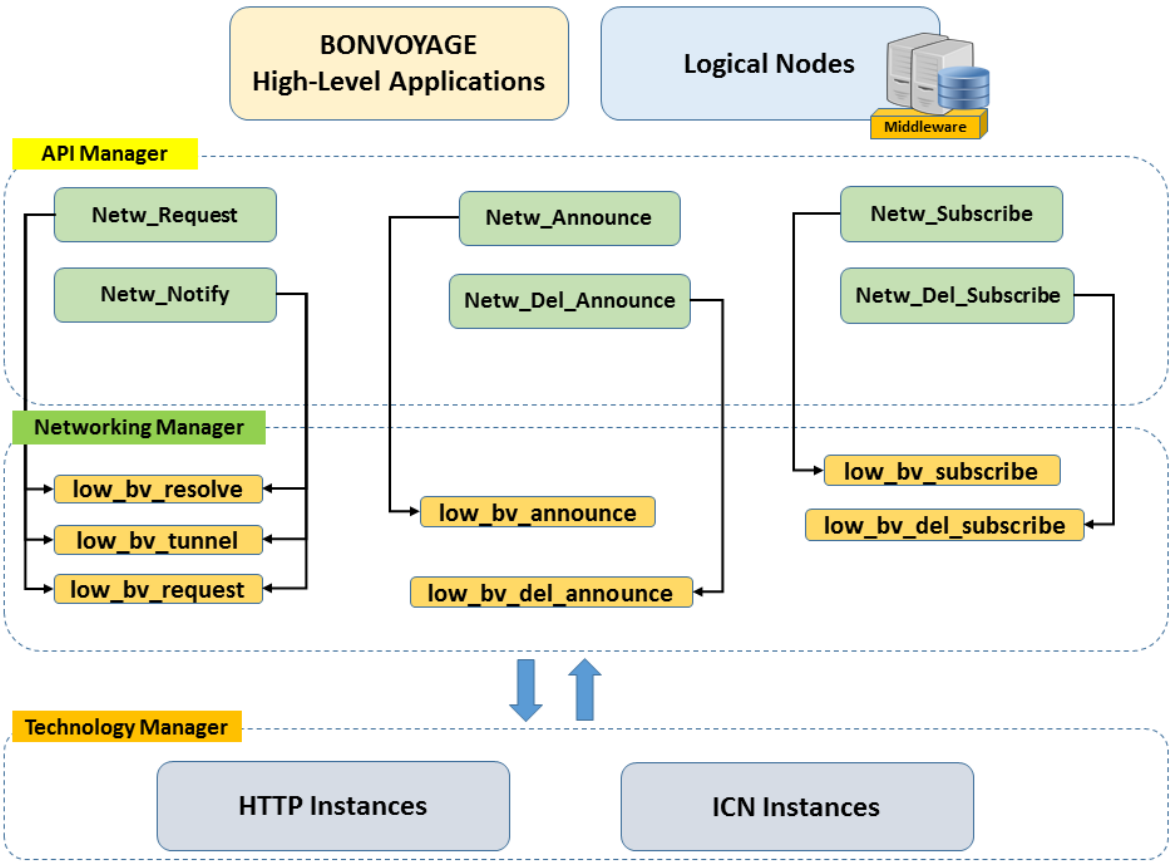

Figure 12: Technical details for the Internames Service Layer

With specific reference to what described in *Deliverable 3.1 - Networking*, a technical description of the Internames Service Layer is reported in Figure 12. It is possible to observe that it integrates three main modules: (i) the *API manager*, that offers the set of APIs to the application, (ii) the *networking manager*, which implements atomic networking operations, and (iii) the *technology manager*, hosting technology-dependent instances able to process messages coming from the network realms (i.e., IP or ICN).

The main rationale characterizing the developed middleware is described in what follows:

- the high level application interacts 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 execution 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:

- **Netw_Request**. As is has been previously recalled, requests are the initial 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. 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_bv_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_bv_resolve*). Then it executes the *low_bv_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_bv_resolve*); then, *low_bv_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.

- **Netw_Announce**. It is adopted by a Data Producer to announce the availability of a content in the BONVOYAGE system. It receives in input the name of the content to be announced and provides in output a message of confirmation. This API executes a single atomic networking operation, for instance, *low_bv_announce*. It is meant for sending the announcement message to the reference IRN logical node.

- **Netw_Del_Announce**. This operation is used by the Data Producer to communicate that it can no longer provide any content for a specific name. The message is sent to

the reference IRN and, in this specific case, the atomic networking operation performed is *low_bv_del_announce*.

- **Netw_Subscribe**. This operation is used by the Data Consumer to make a subscription to a specific content available in the platform. BONVOYAGE receives the content name as an input and provides an output confirmation message. In a similar way to what happens in the previously described case, a single atomic operation is required: in particular, it is a *low_bv_subscribe*. This operation is supposed to sends the subscription request to the reference IRN logical node. Afterwards, the network is aware that the subscription has been done and information provisioning is made possible.

- **Netw_Del_Subscribe**. This operation is used by the Data Consumer for eliminating a subscription. The message is sent and processed by the IRN. The atomic networking operation involved in this case is *low_bv_del_subscribe*.

- **Netw_Notify**. Notification is performed as soon as new contents or data are available within the network. The IRN logical node uses this API to communicate the specific content availability to Data Consumers as soon as a new, or updated, content is available. The function receives three parameters: the content name, the information about the realm to which the subscriber belongs to, and the protocol in use in that specific realm. Just like the other API, it has an output that is a confirmation message. The Netw_Notify requires the execution of one atomic networking operation, that is low_bv_notify.

Table 7 summarizes the details related to the APIs used for both request-response and publish-subscribe communication schema.

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 Table 8. For what concerns the IP realm, it is assumed that messages are exchanged by using HTTP. Without loss of generality, for the ICN realm, the 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 relationship. In this scenario two types of messages are admitted, that are *INTEREST* and *DATA*; a user may ask 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.

A more detailed description has been provided in *D3.1 - Networking*.

API	Atomic operation triggered by the API	Description
Netw_Request	low_bv_request low_bv_resolve low_bv_tunnel	**Role**: requests a travel-centric information or participatory sensing data identified by a given name. **Input**: name or identifier (output of Netw_Map). **Output**: location (next hop) and communication protocol
Netw_Announce	low_bv_announce	**Role**: communicate the availability of a new content, or an updated version of it. **Input**: content name. **Output**: confirmation message
Netw_Del_Announce	low_bv_del_announce	**Role**: cancel the availability of a content. **Input**: content name. **Output**: confirmation
Netw_Subscribe	low_bv_subscribe	**Role**: communicate a subscription to a content. **Input**: content name. **Output**: confirmation
Netw_Del_Subscribe	low_bv_del_subscribe	**Role**: cancel a subscription to a content. **Input**: content name **Output**: confirmation
Netw_Notify	low_bv_notify	**Role**: notify the availability of a new version of the content to subscribed users. **Input**: content name. **Output**: confirmation

Table 7: Atomic networking operations

API	Atomic operation	Realm	Message
Netw_Request	low_bv_request	IP	GET HTTP [IP_ADDR]:[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]
Netw_Announce	low_bv_announce	IP	POST HTTP [IRN_IP]:80/bv/announce, with the body set to n2n://[realm_ID]/[NAME]
		NDN	INTEREST /bv/announce/n2n://[realm_ID]/[NAME]
Netw_Subscribe	low_bv_subscribe	IP	POST HTTP to [IRN_IP]:80/bv/subscribe, with the body set to n2n://[realm_ID]/[NAME]/[proxy_IP]
		NDN	INTEREST /bv/subscribe/n2n://[realm_ID]/[NAME]
Netw_Notify	low_bv_notify	IP	POST HTTP to [proxy_IP]:80/bv/notify, with the body set to n2n://[realm_ID]/[NAME]
		NDN	DATA related to /bv/subscribe/n2n://[realm_ID]/[NAME]

Table 8: Messages generated in IP and NDN network realms.

3.2 Front-end APIs and Data format

In the conceived system, Data Consumer and Data Producer do not interact directly with Internames. They, instead, establish a connection with dedicated nodes (namely, Consumer Proxy and Producer Proxy, respectively) that act as an interface between the users of the BONVOYAGE system and the underlying network infrastructure.

Focusing the attention on Data Producer and Producer Proxy, the following main functionalities were implemented in the BONVOYAGE Communication System:

- Data Producer initiates a WebSocket connection, thus establishing an asynchronous and bidirectional comunication with the Producer Proxy;

- Data Producer announces the availability of real-time data to the Producer Proxy and becomes a source for these data, till the websocket connection is closed. According to what has been previously described, announced data refer to one or more name/topic. For each of them, two different contents must be available: initial contents (also referred to as INIT) and updated contents (namely UPDATE). Initial contents are those that must be delivered to the consumer as soon it joins the platform. This first delivery phase is aimed at providing both static data and initial ones. Both of them are related to initial values but they cover values of the contents that are not going to change (frequently) over time. Update contents, instead, include any modification to data currently produced by the Data Producer. Front-end APIs used to communicate these data are: **Producer_Publish_Init** and **Producer_Publish_Update**. In details, **Producer_Publish_Init** is used to publish the initial content of a generally dynamic publication. It accepts the name of the publication and the publication itself. Since it has been created as a boolean function, it returns "*true*" if the publishing was successful, or "*false*" if something went wrong. In a similar way, **Producer_Publish_Update** is used to publish an update of existing publication, it accepts the name of the publication and the new publication, returning true if the publishing was successful, or false if something went wrong. It is important to remark that data are locally stored on the producer's side (this operation is transparent for the end user, but it is implemented by the front-end APIs). Furthermore, both APIs must be called sequentially every time the content is generated, to properly trigger the notification process. Therefore, when the websocket connection between Data Producer and Producer Proxy is closed, data will be no longer available in the entire system. However, if in-network caching is enabled in the intermediate border routers, a local copy of data can be temporarily available for Data Consumers, even if the Data

Producer is not connected;

- Once the Producer Proxy receives a publication message, it announces the availability of the (new version of the) content to the BONVOYAGE Communication System;

The interactions between Data Consumer and Consumer Proxy have been defined in a similar way. For these two entities, the following main functionalities have been implemented in the BONVOYAGE Communication System:

- Data Consumer initiates a WebSocket connection with the Consumer Proxy;
- Data Consumer communicates to the Consumer Proxy its interest to collect real-time data belonging to a set of names/topics. It is worth noting that the list of names/topics is obtained during the discovery procedure done by querying, for instance, the OGB Discovery Service (which is out of the scope of the present section). It uses the front-end API namely **Consumer_Subscribe**, which receives in input the list of names/topics and triggers the subscription process developed by the Consumer Proxy;
- Once the Consumer Proxy receives the list of names sent by the consumer, it retrieves the corresponding initial contents and makes a subscription request to the BONVOYAGE Communication System (through which it will be able to receive future updates);
- Data Consumer may be interested to retrieve only the initial content belonging to a set of names/topics, without making a subscription for such names: the front-end **Consumer_Request** API is designed for this purpose. Once the Data Consumer calls the aforementioned API, the Consumer Proxy immediately retrieves requested contents by performing the Request-Response interaction.

To implement data exchange functionalities, a standard format has been created. Data Consumers and Data Producers communicate with the proxies of the Internames infrastructure exchanging messages in a precise format, for instance they are JSON messages. They are made up by three fields (see Figure 13):

- **Header**. This field contains general information about the message (e.g., the name of the publication the message is carrying out). This part of the message is mandatory in order to disambiguate communications and properly address the message to the user/consumer/application which asked for it;
- **Type**. This field gives specific information about the type of message. This can either contain logging information or real data. So, this field aims at providing information about the usage of the retrieved data;

- **Content**. In it, the real content of the message is communicated.

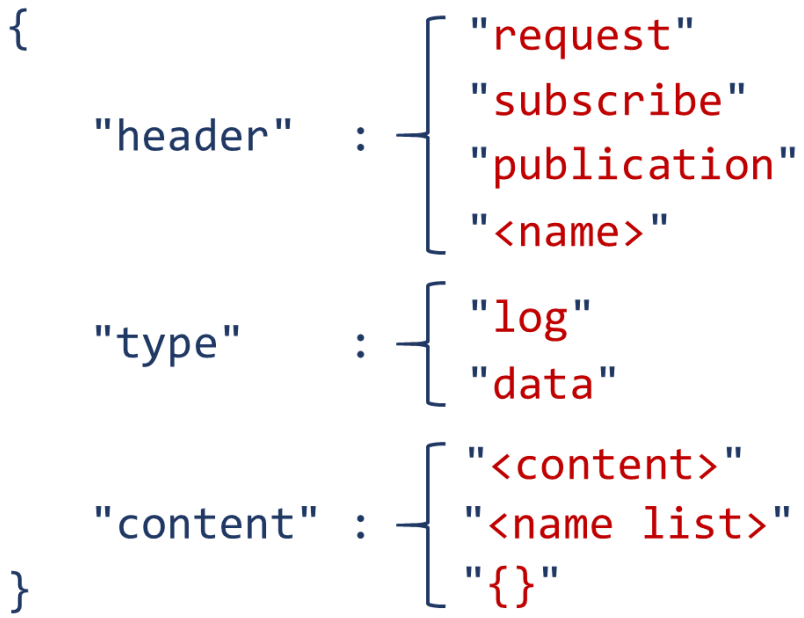

Figure 13: JSON messages structure

The JSON notation has been chosen as it is a lightweight format meant for efficient data exchange. This notation is a widely adopted solution that enables advanced analysis functionalities. In fact, its notation eases both human readability and machine elaboration, for instance, message parsing and generating. Moreover, JSON data format lowers the overhead connected to processing activities which are mandatory to elaborate other standard format files, e.g., XML. For these reasons, using this format for the message, together with the JSON data notation, represents an optimal solution for message exchange between network nodes to grant efficient asynchronous and decoupled communications. It is worth noting that messages can be of different nature and the core network do not discriminate between them: control information and data messages dispatched do not imply any differences in delivering throughout the network.

4 Publish-Subscribe functionalities

The BONVOYAGE Communication System relies on Publish-Subscribe primitives to grant asynchronous message exchange. Publish-Subscribe primitives require the execution of additional functionalities, like publication, subscription, notification and Request-Response. To describe the main facets related to these communication scheme, the following scenario is taken into account: a Data Consumer and a Data Producer are connected to the BONVOYAGE platform through the respective Proxy. The Data Producer and the Data Consumer use the related front-end APIs to publish and subscribe to a content of interest, respectively.

All the functionalities implemented in the BONVOYAGE Communication System imply messages exchange between the Involved Entities. Accordingly to what stated until this point, it is important to remark that not all the interactions involve all the entities. For the sake of clarity, Figure 14 explains the related details. In particular, it gives explanation of the entities involved by each interaction. Moreover, a precise indication of which API in specifically involved for each interaction is given.

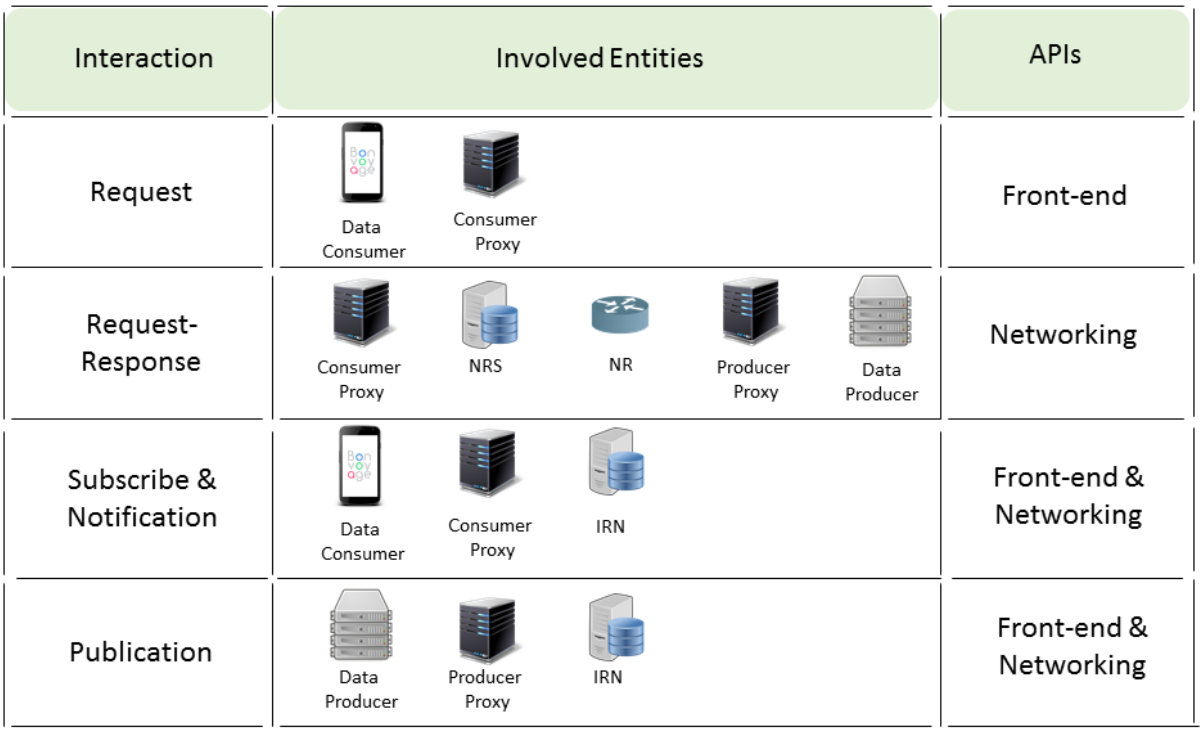

Figure 14: Publish-Subscribe interactions

A further insight in such interactions is given in Figure 15, where the Involved Entities for each communication activity is highlighted. For the sake of completeness, Publish-Subscribe interactions are described with Figure 16, where the entities involved for each communication activity are highlighted.

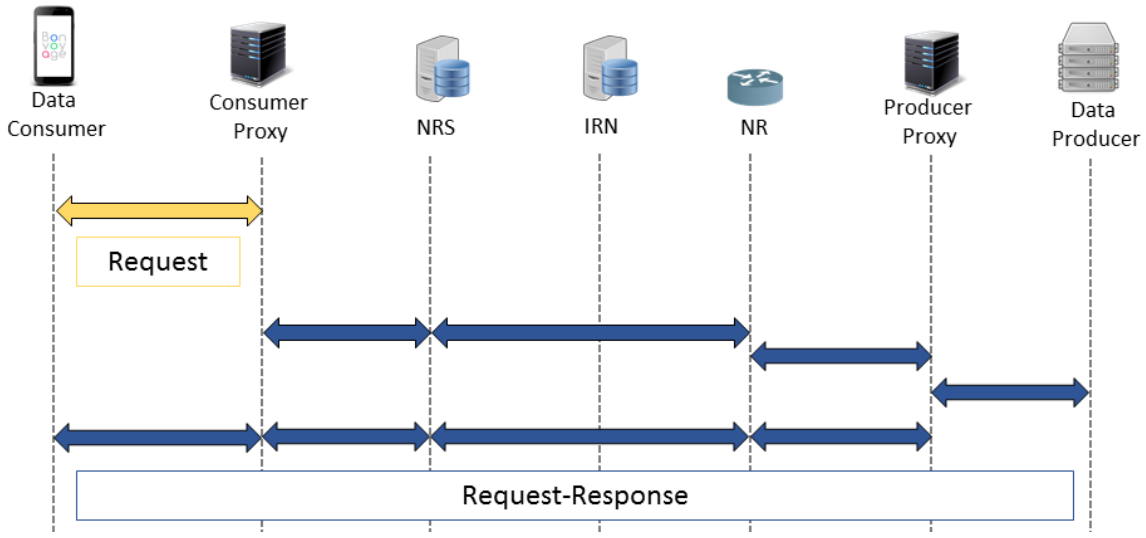

Figure 15: Request-Response interactions - Items involved

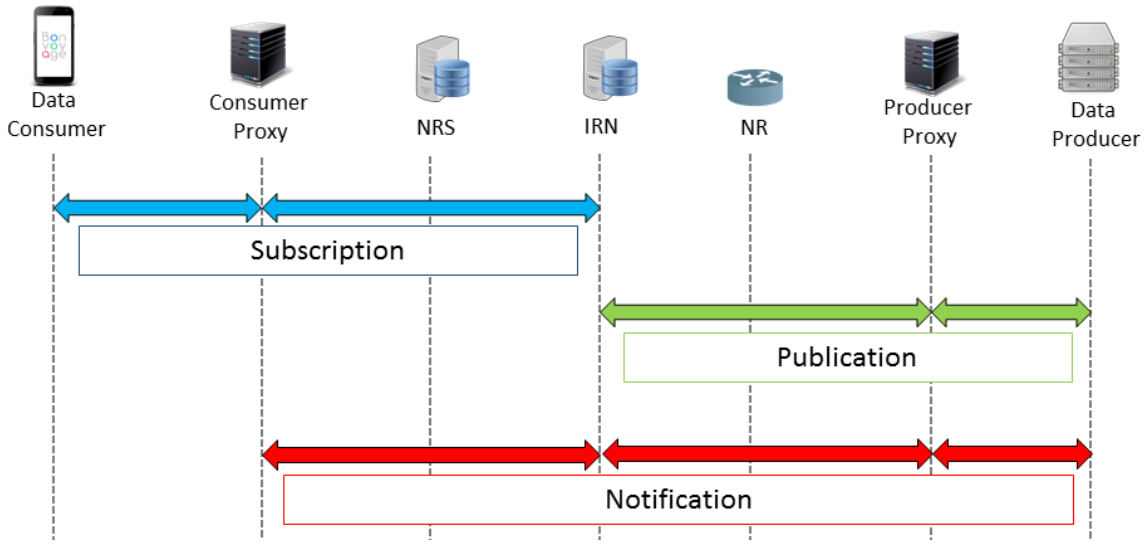

Figure 16: Publish-Subscribe interactions - Items involved

4.1 Request of a content

The request of a content is the interaction that allows a Data Consumer to retrieve only the initial travel-centric data belonging to a set of name/topics of interest, without making a subscription for such names. As well as for Data Producer, also the Data Consumer is not directly involved in Internames. In fact, the interaction is mediated by the dedicated Consumer Proxy, which offers a standardized interface with the rest of the BONVOYAGE platform, through the lightweight WebSocket technology. The latter is used in conjunction with JSON format message encoding and a set of dedicated front-end APIs. For contents retrieval, the Consumer Proxy performs the Request-Response interaction with the rest of the BONVOYAGE platform, as described above.

The messages exchanged while performing a request for a content between Data Consumer, Consumer Proxy and the rest of the BONVOYAGE platform, are summarized in Figure 17.

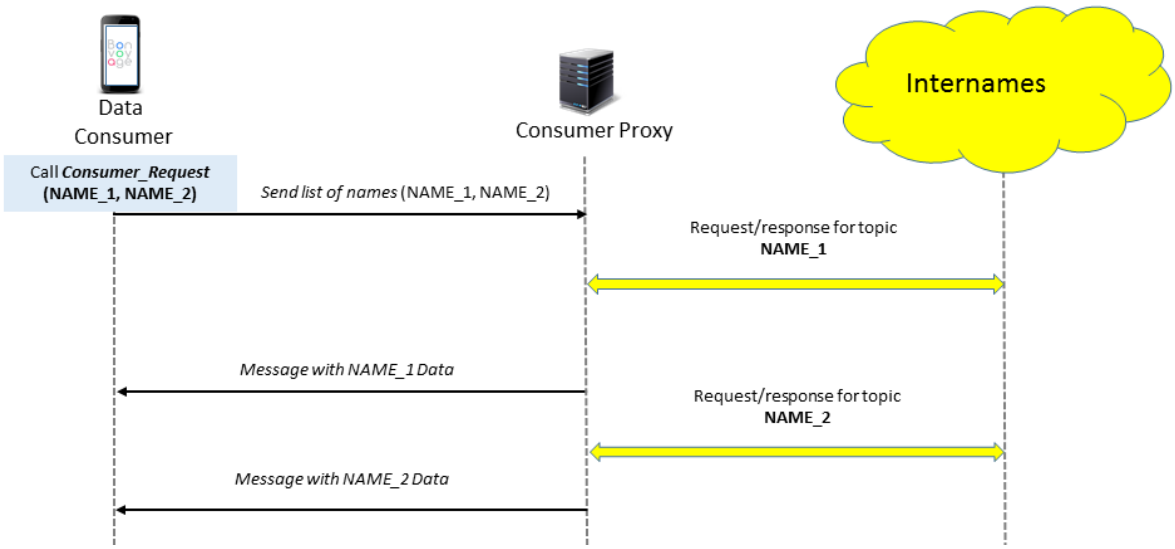

Figure 17: Request of a content in BONVOYAGE

In details, the steps are described below:

1. At first, the Data Consumer opens a WebSocket session with the reference Consumer Proxy (not depicted here);
2. The Data Consumer calls the *Consumer_Request* API. As a parameter, the list of names is given. In this example case, the list contains two names, referred to as **NAME_1** and **NAME_2**;
3. The *Consumer_Request* API encodes the list of names in a JSON message, and sends it to the reference Consumer Proxy;

4. For each name, the Consumer Proxy immediately retrieves the initial content, by performing the Request-Response interaction (described later on in this Section).

4.2 Publication of a content

In BONVOYAGE, the publication of content is the interaction that allows a generic Data Producer to publish new (or updated) data, under a specific topic of interest. As described in previous Sections, the Data Producer is not directly involved in Internames. Instead, it interacts with the dedicated Producer Proxy through a specific front-end API, exploiting the asynchronous and lightweight WebSocket technology. In this case, the underlying communication at middleware layer involves the Internames Rendezvous Node, which is able to properly manage the announcements of a content to the BONVOYAGE platform. To this end, the *Newt_Announce API* is called by the Producer Proxy.

Figure 18 shows the sequence diagram of messages exchanged between Data Producer, Producer Proxy and Internames Rendezvous Node.

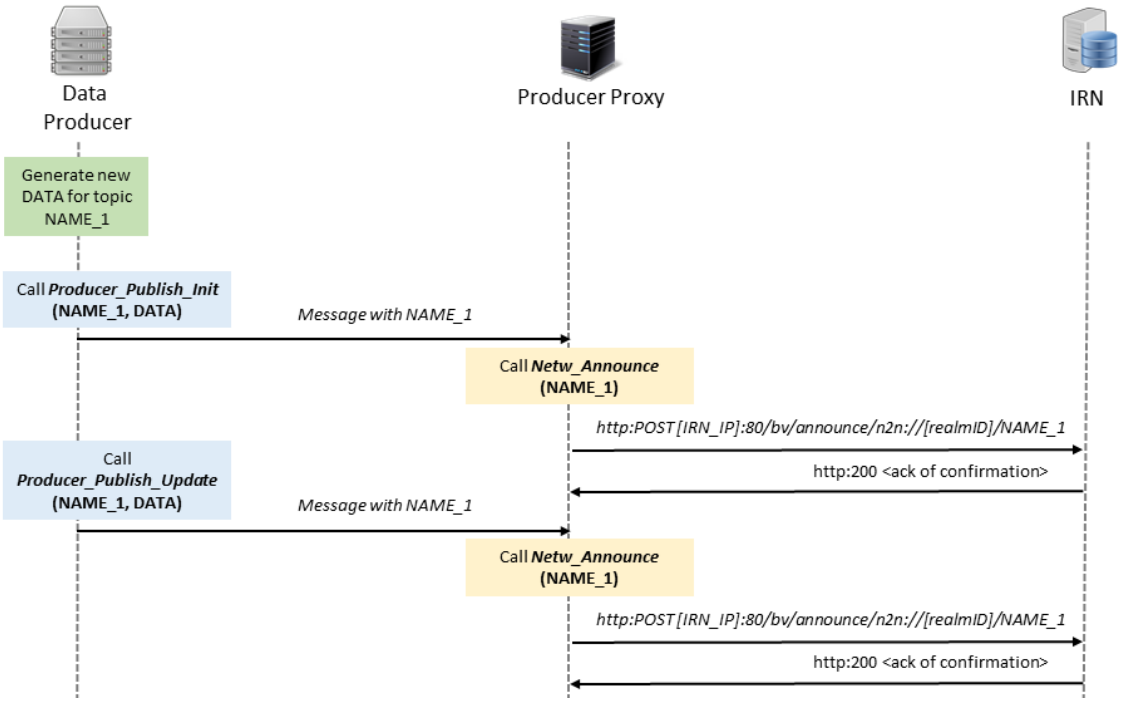

Figure 18: Publication of content in BONVOYAGE

In details, the steps are described below:

1. At first, the Data Producer opens a WebSocket session with the reference Producer Proxy (not depicted here);

2. When the Data Producer generates a new data under a specific topic (*NAME_1* in this

example), it calls the *Producer_Publish_Init* front-end API to set the initial content for the topic of interest, with the name and the data as a parameters;

3. The *Producer_Publish_Init* stores the data in a dedicated folder, according to its filesystem structure. Then, it sends a JSON message structured with the aforementioned format to the Producer Proxy;

4. After receiving the message, the Producer Proxy calls the *Netw_Announce* network API in order to advertise the availability of a new data for the topic of interest;

5. According to the underlying communication technology, the Internames Service Layer translates this high-level request into a specific message. The message is delivered to the reference Internames Rendezvous Node through HTTP, by using the POST method:

HTTP: POST

[IRN_IP]:80/bv/announce/n2n://[realm IP]/[NAME]

,

where the field [NAME], in this case, is set to *NAME_1*.

6. The IRN will answer with an HTTP status code of confirmation;

7. The Producer Proxy calls the *Producer_Publish_Update* front-end API to set the punctual update for the topic of interest, using the name and the data as parameters;

8. To announce the availability of the punctual update, the steps from 4 to 6 are repeated.

4.3 Subscription to a content

The subscription to a content is the interaction that allows a Data Consumer to collect travel-centric data belonging to a set of names/topics of interest, in order to retrieve the initial contents and punctual updates (if any). In a similar way to what described the request of a content, in this case the Data Consumer communicates with the Consumer Proxy through a simplified interface, exploiting the WebSocket technology and using dedicated front-end APIs.

As well as for publication, the underlying communication at middleware layer involve the Internames Rendezvous Node. The latter, in fact, is able to manage the subscription requests for contents available in the BONVOYAGE platform. To this end, the *Netw_Subscribe* API is called by the Consumer Proxy. In particular, this happens for each name/topic of interest. The main

functionality of the Consumer Proxy is the aggregation of subscription requests. Internally, it creates a subscription list that maps each name with the Data Consumers subscribed to it. Indeed, given a message with the list of names, for each name, it verifies if the related subscription list is empty. In this case, it performs the *Netw_Subscribe* to the Internames Rendezvous Node. Otherwise, it just adds the new Data Consumer in the subscription list.

Figure 18 shows the sequence chart of messages exchanged between Data Consumer, Consumer Proxy and Internames Rendezvous Node. To explain the aggregation process, a use case with two Data Consumers is taken into account. The subscription lists for each name are supposed to be initially empty.

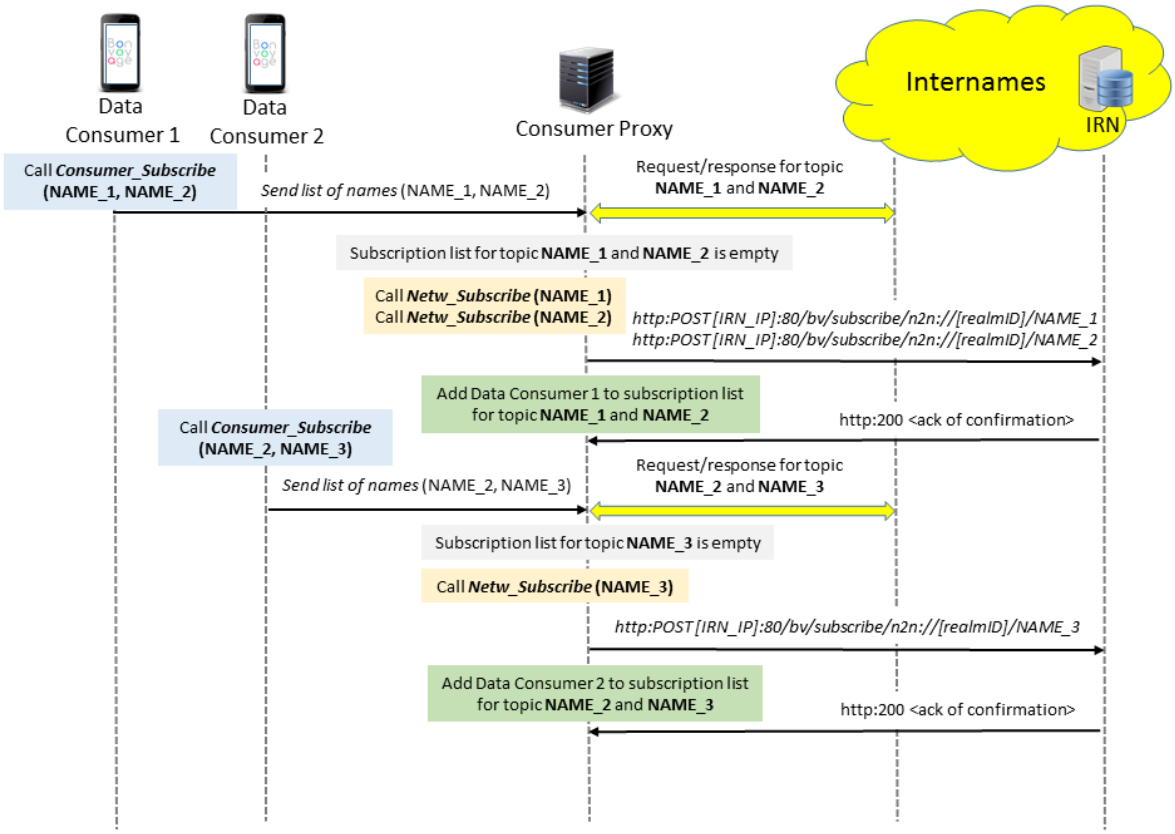

Figure 19: Subscription to a content in BONVOYAGE

The subscription procedure details are described below:

1. At first, each Data Consumer opens a WebSocket session with the reference Consumer Proxy (not depicted here);

2. The "Data Consumer 1" calls the *Consumer_Subscribe* API, giving the list of names as a parameter. In this case, the list contains two names **NAME_1** and **NAME_2**;

3. The *Consumer_Subscribe* API encodes the list of names using the JSON format notation for the message, and sends it to the reference Consumer Proxy;

4. The Consumer Proxy immediately retrieves the initial contents associated to the names in the list, by triggering the Request-Response interaction;

5. For each name, it verifies if the related subscription is empty. In this case, both lists for **NAME_1** and **NAME_2** are empty, so the Consumer Proxy calls the *Netw_Subscribe* twice by passing the aforementioned names;

6. According to the underlying communication technology, the Internames Service Layer translates these high-level requests into a specific message. The messages are delivered to the reference Internames Rendezvous Node through the HTTP protocol, by using the POST method:

   HTTP: POST

   [IRN_IP]:80/bv/subscribe/n2n://[realm IP]/[NAME]

   ,

   where the field [NAME], in this case, is set to *NAME_1* and *NAME_2*, respectively.

7. The Consumer Proxy adds "Data Consumer 1" to the subscription lists for both **NAME_1** and **NAME_2**;

8. The Internames Rendezvous Node will answer with an HTTP packet of confirmation;

9. The "Data Consumer 2" calls the *Consumer_Subscribe* API, with the list of names as a parameter. In this case, the list contains the names **NAME_2** and **NAME_3**;

10. The Consumer Proxy immediately retrieves the initial contents associated with the names in the list, by triggering the Request-Response interaction;

11. For each name, the Consumer Proxy verifies if the related subscription is empty. In this case, **NAME_3** is empty. Conversely, **NAME_2** already contains an entry, with reference to the "Data Consumer 1" (already subscribed in the above steps). As a consequence, the Consumer Proxy will call the *Netw_Subscribe*, but only for **NAME_3**;

12. After sending a message in a similar fashion to what happened at step 6, the Consumer Proxy adds "Data Consumer 2" to the subscription lists for both names, **NAME_2** and **NAME_3**;

4.4 Notification process

The notification process is the interaction that allows Data Consumers to retrieve punctual updates related to the topics to which they are already subscribed. When the Internames Rendezvous Node realizes the availability of a new version of a travel-centric content, it sends a notification to all the subscribers by using the *Netw_Notify* API. The Consumer Proxy, as well as for subscription requests, aggregates the notification delivery for all subscribed users. In order to receive the punctual update after the notification, the Data Consumer needs to keep alive the WebSocket session (that it has opened before) after sending the list of names. The Consumer Proxy performs the Request-Response interaction for each name of interest.

The messages exchanged during the notification process between Internames Rendezvous Node, Consumer Proxy and Data Consumer are summarized in Figure 20. To show the aggregation process for notifications, a use case with two Data Consumers is taken into account. The Data Consumers are supposed to be subscribed for a topic called **NAME_2**.

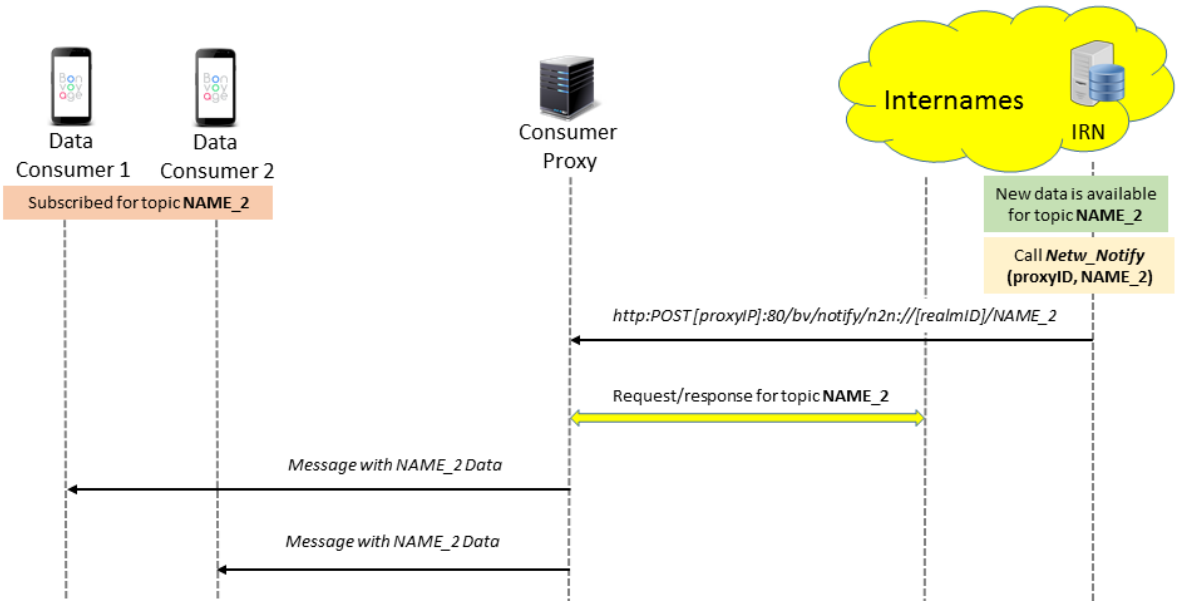

Figure 20: Notification in BONVOYAGE

In details, the steps are described below:

1. At first, the Internames Rendezvous Node realizes the availability of a new content under the topic **NAME_1**. Then, it calls the *Netw_Notify* API in order to notify the reference

Consumer Proxy, by passing the proxy IP address and the name as parameters;

2. According to the underlying communication technology, the Internames Service Layer translates this high-level request into a specific message. The messages are delivered to the reference Consumer Proxy through the HTTP protocol, by using the POST method:

HTTP: POST

[PROXY_IP]:80/bv/notify/n2n://[realm IP]/[NAME]

3. After receiving the message, the Consumer Proxy performs the Request-Response interaction, retrieves the punctual update and sends it to all the Data Consumers subscribed for the topic of interest via the previously established WebSocket connection (not shown here).

4.5 Request-Response communication scheme

The Request-Response communication scheme allows the Consumer Proxy to retrieve both the initial data and the punctual update, for a topic of interest in the BONVOYAGE platform. The Name Resolution Service and Named Router logical nodes, for each realm, are involved in this interaction. Their role is to handle the routing and forwarding operations across heterogeneous realms. Figure 21 shows the sequence diagram of messages exchanged between Proxies and logical nodes.

In details, the steps are described below:

1. The Consumer Proxy selects a content of its interest, for example:

n2n://[realm_IP]/[NAME]

and initiates the Request-Response data exchange through the *Netw_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 Name Resolution Service node:

HTTP: GET

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

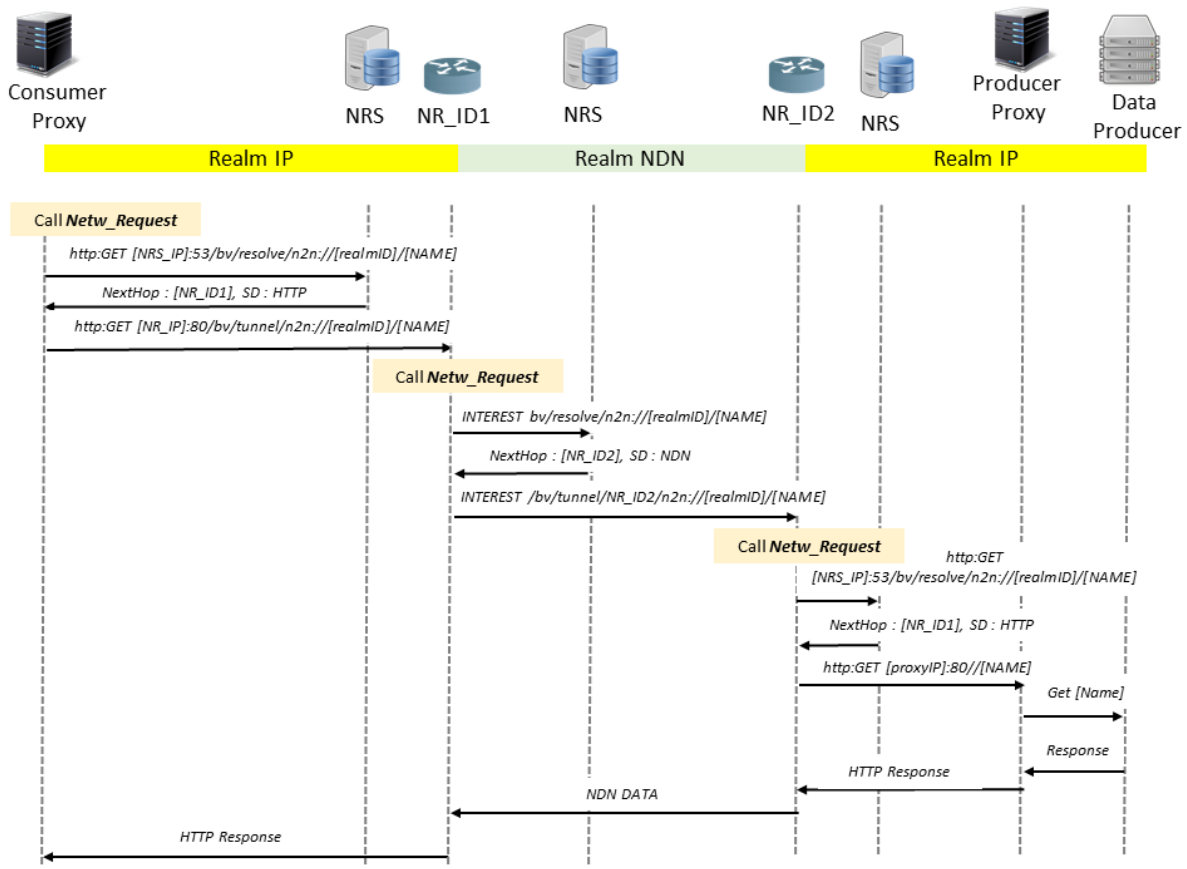

Figure 21: Request-Response in BONVOYAGE

2. The reference NRS node answers with routing details:

HTTP: 200

⟨next hop=NR_ID1; protocol=IP⟩

3. The Internames Service Layer instance of the Consumer Proxy executes the *low_bv_tunnel* atomic networking operation that forwards the request to the Named Router NR_ID1. To this end, it sends a GET HTTP message to the reference Name Resolution Service node:

HTTP: GET

[NR_ID1]:80/bv/tunnel/n2n://[realm_IP]/[NAME]

4. NR_ID1 asks the NRS within NDN realm about how to resolve the name of the content of interest for the Consumer Proxy. The message sent is:

NDN: INTEREST

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

5. The answer comes from the NRS within the NDN realm and indicates the following hop to NR_ID2, the Named Router between NDN and IP realms:

NDN: DATA

⟨next hop=NR_ID2; protocol = NDN⟩

6. NR_ID1 delivers the request to NR_ID2 through an INTEREST message:

NDN: INTEREST

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

7. The message is processed by the Named Router, that re-initiates the Request-Response data exchange through the *Netw_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]

8. The reference NRS node answers with routing details:

HTTP: 200

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

9. At this step, the Internames Service Layer instance realizes that the requested content is available in the IP realm to which it is directly attached. Therefore, from the realm-based naming structure it extracts the [NAME] that can be used within the IP realm. Afterward, it sends a GET HTTP towards the IP address of the reference Producer Proxy:

HTTP: GET

[proxy_IP]:[port]/[NAME]

10. The Producer Proxy encodes the request according to the JSON message notation and sends it to the Data Producer through a WebSocket connection;

11. The Data Producer encodes the requested travel-centric data in a JSON message and sends it back to the Producer Proxy, through the same WebSocket connection;

12. The Producer Proxy answers with the requested travel-centric data:

HTTP: 200

⟨Requested Data⟩

13. The Named Router NR_ID2 processes the messages and stores the received content within a DATA packet to forward in the NDN realm, back to the Consumer Proxy:

NDN: DATA

⟨Requested Data⟩

14. The Named Router NR_ID1 processes the messages and stores the received content within HTTP message to forward in the IP realm, back to the Consumer Proxy:

HTTP: 200

⟨Requested Data⟩

5 Experimental testbeds

The main functionalities of the BONVOYAGE Communication System are investigated here with two experimental testbeds. They are aimed at demonstrating two different use cases, when implementing Publish-Subscribe functionalities.

The first testbed aims at demonstrating the adoption of networking APIs on the Internames Service Layer middleware within the ICN-IoT architecture.

On the other hand, the second testbed is in charge of showing the behavior pf the whole BONVOYAGE Communication System in a concrete implementation.

5.1 Testbed focusing on networking APIs

ICN-IoT [11] aims at designing a unified IoT platform based on ICN principles. It was conceived by the Information-Centric Networking Research Group (ICNRG), which belongs to Internet Research Task Force (IRTF).

ICN-IoT and BONVOYAGE Communication System share, at network layer, the same functionalities. In fact, in both cases, the communication bus is meant for disseminating contents across a heterogeneous network through Request-Response and Publish-Subscribe communication schema. Therefore, the testbed presented herein demonstrates how the outcomes of the BONVOYAGE project (specifically those coming from WP3) can be applied also to contexts beyond the Intelligent Transport System field.

A heterogeneous network made up by two different realms (e.g., ICN and IP), connected to each other with a dedicated border router, is considered. The IoT domain exposes its resources through a gateway, attached to the IP realm. In the implemented ICN-IoT solution, in fact, unique names are used to identify contents that are resources made available by IoT devices. As a consequence, they do not contain any reference to Data Producer. However, to grant system interoperability, names can be structured by defining a unique namespace. When a Data Consumer wants to retrieve a given content, it issues a request to the ICN-IoT middleware. The content of the request only contains the corresponding name. The middleware implements all the required networking functionalities (for instance, name resolution, discovery, delivering) to retrieve the requested content.

By leveraging in-network storage/caching and multicast functionalities, natively offered by ICN, ICN-IoT proposes several advantages: (i) data delivery optimization, reduction of (ii) the overall traffic volume, and (iii) the number of requests processed by the IoT devices.

In particular, this aspect is of great interest as such systems are usually constrained in terms of both computational and storage capabilities. Data exchange is based on the receiver-oriented

communication scheme. Accordingly the middleware can realize a seamless support for the mobility, delay tolerant communications, and reliable content delivery over unreliable links.

The deployed testbed reflects the reference scenario depicted in Figure 22. It considers a heterogeneous network, composed by two network realms: IP and NDN. These realms are connected each other through a border router. A single Data Producer belongs to the Internet of Things (IoT) domain and it is attached to the IP realm through a network gateway. The Data Consumer, instead, is installed on a computer attached to the NDN realm. In the simple example described herein, it is assumed that the consumer is interested to get a data generated from the ambient light sensor, available on the board of one constrained device belonging to the IoT domain. Finally, both NRS and IRN are implemented in a centralized fashion and installed on the Named Router. Therefore, they are reachable from both the network realms.

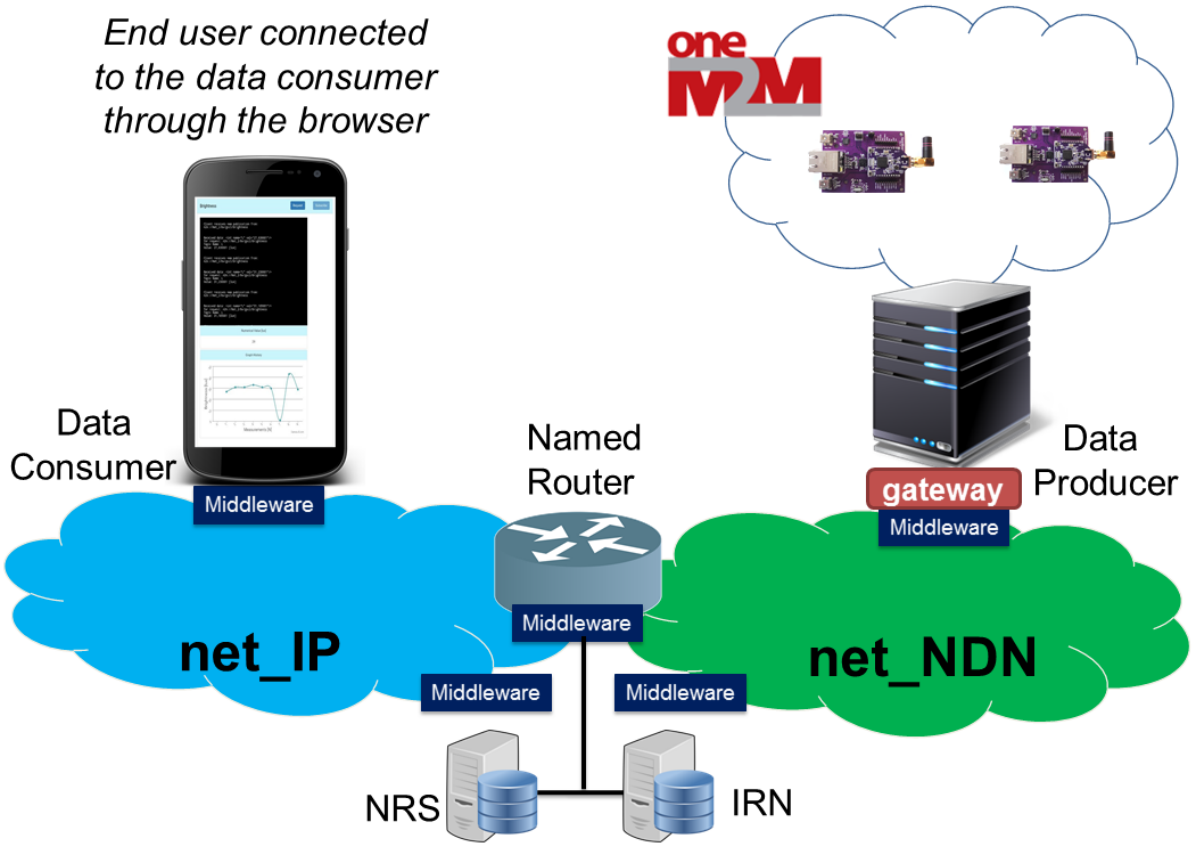

Figure 22: ICN-IoT - Reference scenario implemented in the testbed

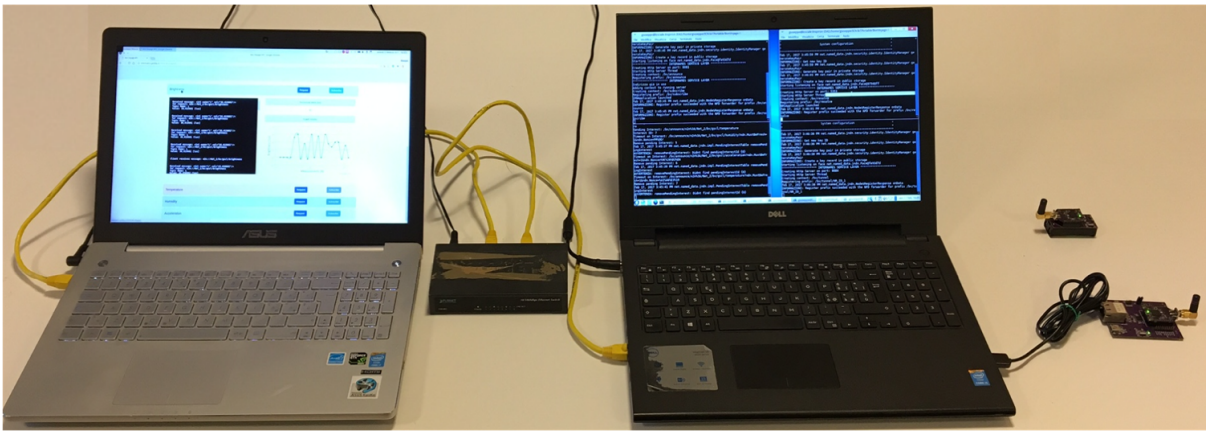

Figure 23: Hardware components forming the developed testbed

With reference to the conceptual representation in Figure 22, the implemented version of the envisioned testbed is depicted in Figure 23. Here, Data Consumer, NDN realm, border router, IRN, and NRS are emulated on a laptop.

The IoT domain is deployed using the OpenMote platform [12]. This IoT device has three main building blocks: (i) the OpenMote-CC2538, (ii) the OpenBase, and (iii) the OpenBattery. The first one is the portion of the constrained device that integrates computational and storage capabilities, together with transceiver functionalities. The CC2538, in fact, is a System-on-a-Chip (SoC) integrated solution that is made of both an MicroController Unit (MCU) and the RF transceiver. The OpenBase module, instead, is used for configuration purposes. In fact, it is used as an interface for the OpenMote-CC2538 when the integrated solution has to be programmed. The OpenBase module also provides a direct connection interface between the IoT node and the laptop. The third module, the OpenBattery, contains a set of sensors to gather data coming from the environment the node is within. In particular, OpenBattery includes three kind of sensors, able to measure temperature, relative humidity, environmental light, and acceleration.

Without loss of generality, the IoT network has two constrained devices: the network coordinator and the remote node. The communication between these nodes is based on the 6tisch protocol suite (that is also different from the conventional TCP/IP stack, commonly used in the current Internet architecture) [13], implemented by the OpenWSN operating system [14].

IoT resources are exposed through the Smart M2M specification [15][16], implemented in the OM2M project (released as an open-source tool by LAAS-CNRS) [17]. OM2M provides a horizontal M2M services platform for the development of services regardless of the underlying network, facilitating interoperability between heterogeneous devices, such as sensors, actuators or data server. To guarantee interoperability between OM2M and Internames, each OM2M

resource is mapped to a unique name in the BONVOYAGE Communication System and that translation is directly handled by the network gateway. The Data Producer entity gathers data from the IoT domain through the network gateway to allow the publication for the rest of testbed, for instance Internames.

The network gateway is installed on the Pandaboard ES platform [18]. It is a low-power and low-cost single-board computer development platform, which integrates Dual-Core 1.2 GHz ARM Cortex-A9 MPCore CPU, 384 MHz PowerVr SGX540 GPU, IVA multimedia hardware accelerator with a programmable DSP, and 1 GB of DDR2 SDRAM, as well as SD Card slot, 10/100 Ethernet, Wi-Fi, and Bluetooth interfaces, output video signal via DVI and HDMI interfaces, and two USB ports. The network gateway is connected to the laptop through an Ethernet cable, which emulates the IP realms.

The aim of this section is to deeply investigate the list of operations executed within the deployed testbed. In this way, it will possible to demonstrate that the end user is able to collect data generated within a remote IoT 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 IoT domain. The aforementioned operations are summarized in Figure 24 and reported below:

- The border router initiates the $/ICN - IoT/tunnel/$ 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 10.0.0.1, port 8080) for the IP realm and a NDN instance for the NDN realm. The NDN instance registers the prefix $/ICN - IoT/tunnel/Router\_ID/$ in the NDN realm, thus being able to receive any request belonging to the *tunnel* service (i.e., inter-realm routing).

- NRS initiates the $/ICN - IoT/resolve/$ 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 10.0.0.1, port 5353) for the IP realm and a NDN instance for the NDN realm. The NDN instance registers the prefix $/ICN - IoT/resolve/$ in the NDN realm, thus being able to receive any request belonging to the *resolve* service.

- IRN initiates both $/ICN - IoT/subscribe$ and $/ICN - IoT/announce/$ services. It is attached to both IP and NDN realms. The technology manager of the designed middleware hosts a HTTP server (IP address set to 10.0.0.1, port 8081) for the IP realm and a NDN instance of the NDN realm. The NDN instance registers the prefixes $/ICN - IoT/subscribe/$ and $/ICN - IoT/announce/$ in the NDN realm, thus being able to receive

any request belonging to the *subscribe* and *announce* services.

- The Data Consumer calls Netw_Subscribe to issue a subscription request for the considered content. The middleware executes the *low_bv_subscribe* atomic operation and sends a HTTP POST message to the IRN, whose body stores the name of the content of its interest.

- The ambient light sensor is stimulated for capturing a variation of the brightness. The gateway devices recognizes the resource update according to the oM2M architecture and calls ICN-IoT ANNOUNCE to communicate the event to the rest of the platform. Therefore, the networking manager executes the *low_bv_announce* and sends a NDN INTEREST to the reference IRN node. After having updated the database, the IRN node answers with a NDN DATA packet of confirmation.

- IRN calls Netw_Notify and notifies the Data Consumer by sending a HTTP POST message, whose body stores the name of the updated content.

- The Data Consumer requests the data through Netw_Request. First, the middleware executes the *low_bv_resolve* atomic operation for contacting the NRS node and getting routing information. NRS answers with the next hop (i.e., the border router) and the communication protocol to use (i.e., IP).

- The middleware instance of the Data Consumer executes the *low_bv_tunnel* atomic operation and sends a HTTP GET packet to the border router, asking for the content of interest.

- The router processes the message and realizes that the requested data is available in the NDN realm to which it is directly connected. Therefore, it forwards the request to the Data Consumer by using a NDN INTEREST packet (the message is generated by the *low_bv_request* atomic operation).

- The network gateway generates the corresponding request, that will be forwarded back to the Data Consumer.

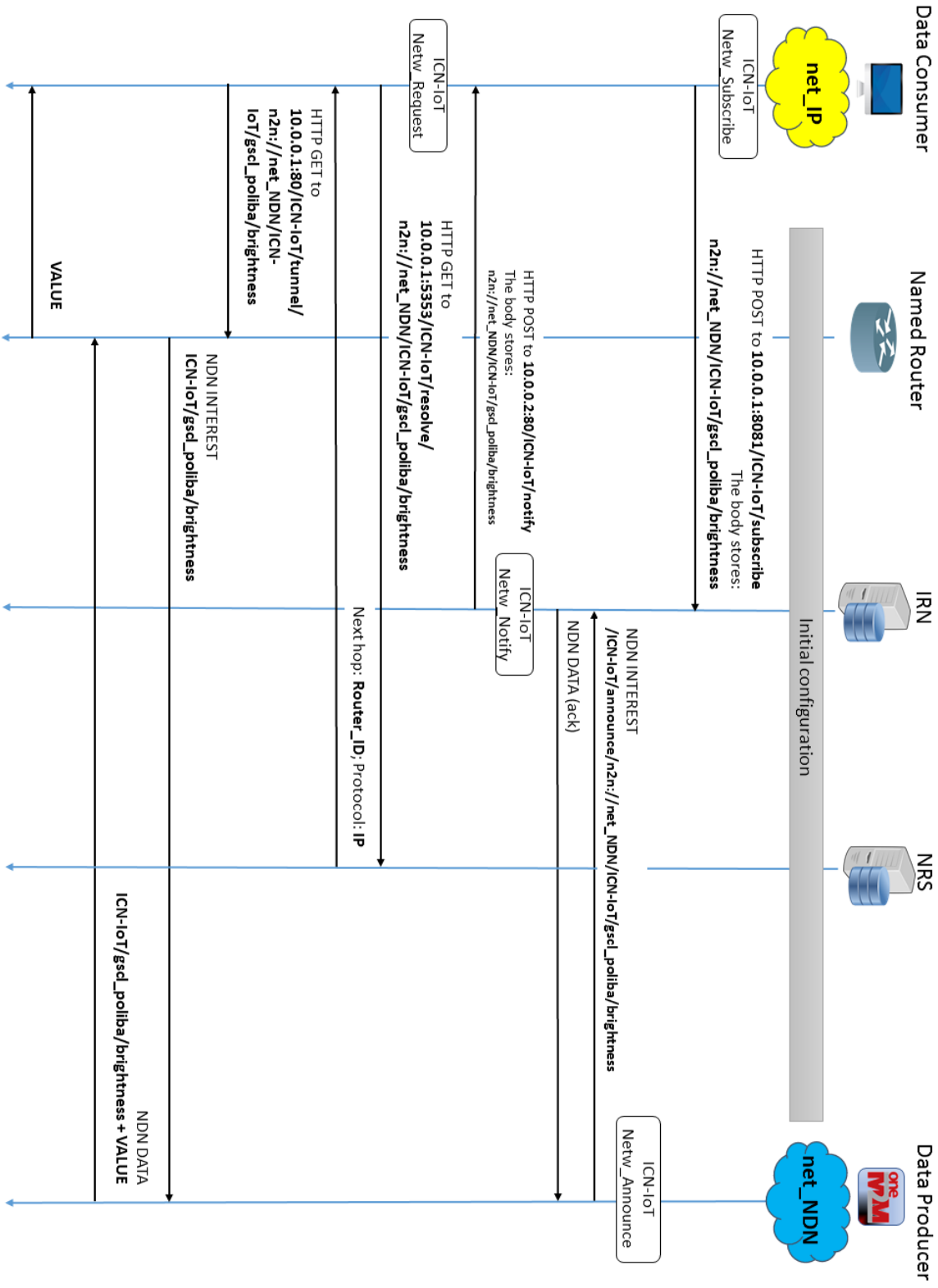

Figure 24: Sequence diagram.

With Figure 25 the web interface for Humidity to the platform is shown. Data logging functionality is exploited thanks to a shell reporting all the values received, by the Data Consumer, after their publication. Indeed, a graphical representation of the acquired data is displayed, as

a function of the number of measurements (12 values in this case).

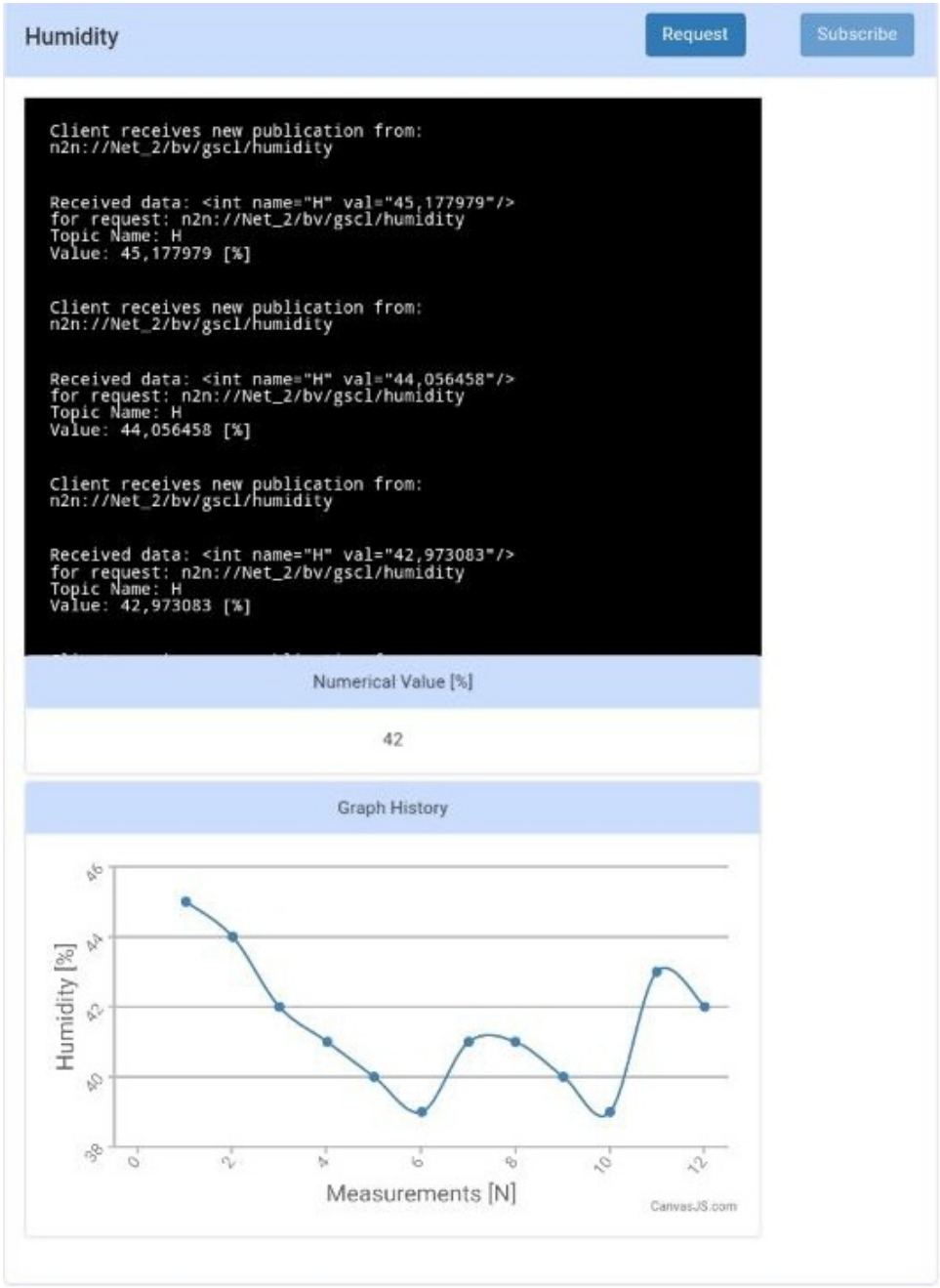

Figure 25: Steady state results for Humidity

With Figure 26 the web interface for Environmental light (namely, Brightness) to the platform is shown. Even in this case, the shell reports all the values received, by the Data Consumer, after their publication. The graph represents acquired data as a function of the number of measurements (9 values in this case).

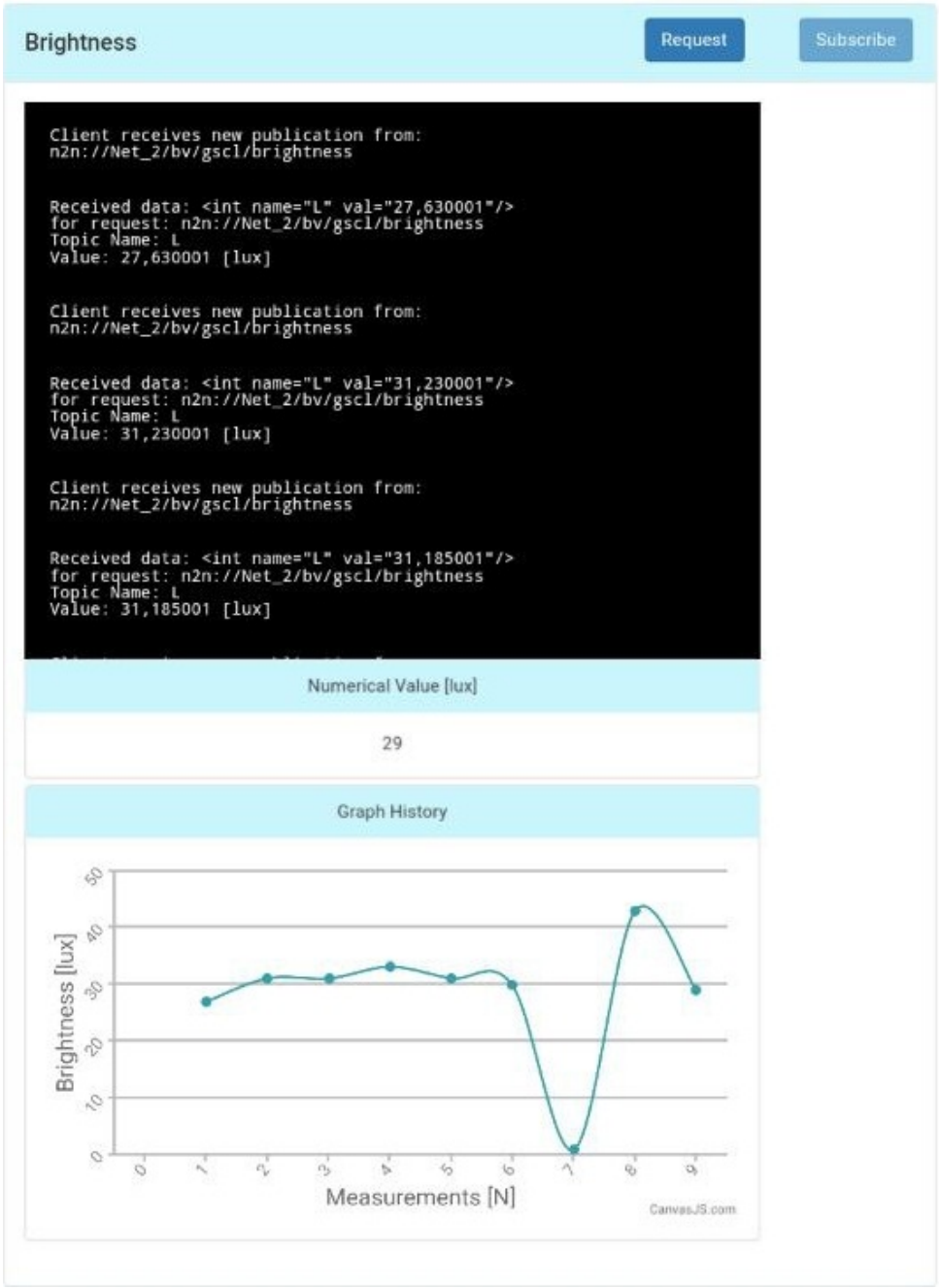

Figure 26: Steady state results for Brightness

With Figures 27 and 28 the web interface shows Temperature and Acceleration, respectively. Their structure is still the same and even in these cases acquired data are displayed with respects to the number of acquired values (6 and 9, respectively).

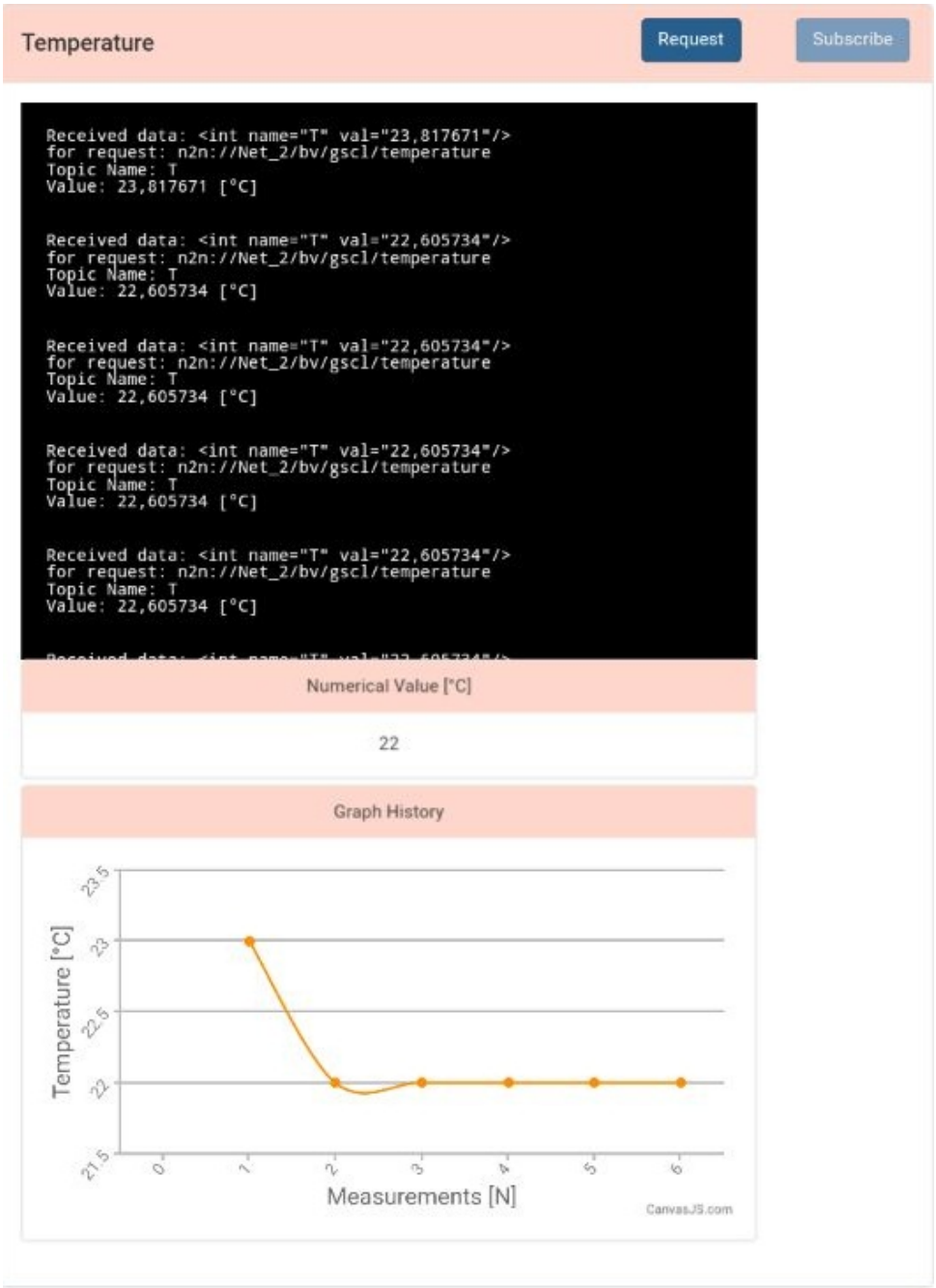

Figure 27: Steady state results for Temperature

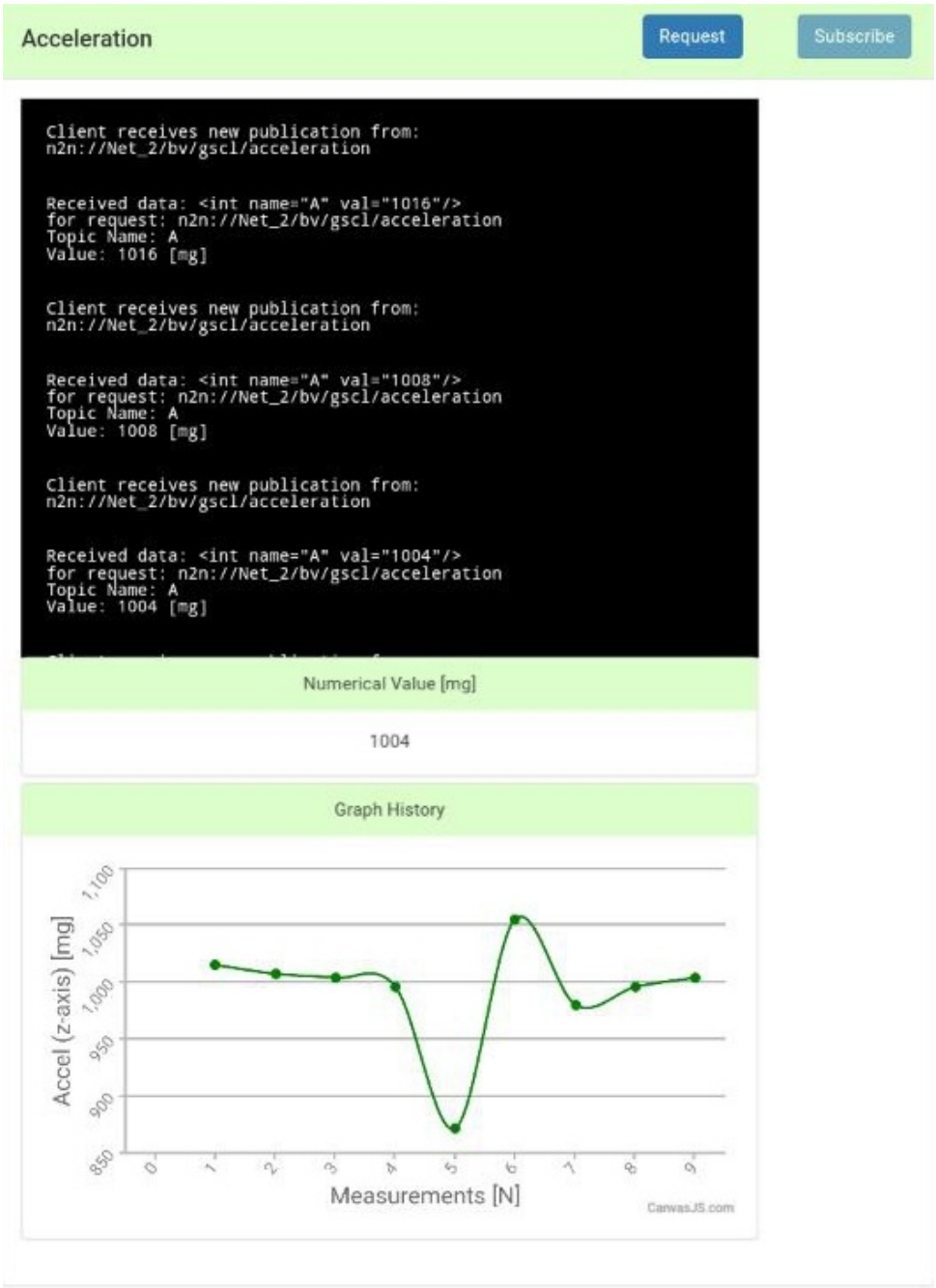

Figure 28: Steady state results for Acceleration

Data coming from the sensor onboard of the constrained device are properly received by the platform and graphically represented as a function of the number of measurements performed. In particular, accordingly to what previously stated, measured variables are Humidity (Figure 25), Environmental light (referred to as Brightness in Figure 26), Temperature (Figure 27, and Acceleration (Figure 28). It is worth noting that such functionalities can be extended to data specifically relevant to the transport domain (e.g. realtime arrival times, etc.)

5.2 Testbed leveraging the whole BONVOYAGE Communication System

In general, the BONVOYAGE Communication System can be used to disseminate any kind of contents. Data Producers, however, must properly arrange their data, thus making their representation in line with the design principles presented in the previous Sections. On the other hand, Data Consumers interested in custom-formatted data, must have knowledge about their representation, to properly parse and process the information in high-level applications.

This section introduces an experimental testbed that focuses on implementing and testing the Publish-Subscribe functionalities, in order to disseminate a customized data by jointly exploiting the Proxies, the APIs and the middleware features exposed so far.

According to the Big Picture of the BONVOYAGE Communication System, a heterogeneous network scenario has been set up, as shown in Figure 29. The implemented entities are:

- Named Routers that interconnect an NDN realm, namely **Net_3** (that acts as Core Network), with two IP realms, namely **Net_1** and **Net_2**, respectively;
- Name Resolution Services for path discovery and routing operations across the aformentioned realms;
- Consumer and Producer proxies connected to the IP realms, that expose WebSocket endpoints to Data Consumers/Producers;
- An instance of Data Producer and Data Consumer that generates formatted data, publishes them in the BONVOYAGE platform, subscribes to the related topic and retrieves the init and updates contents, by using the front-end APIs.

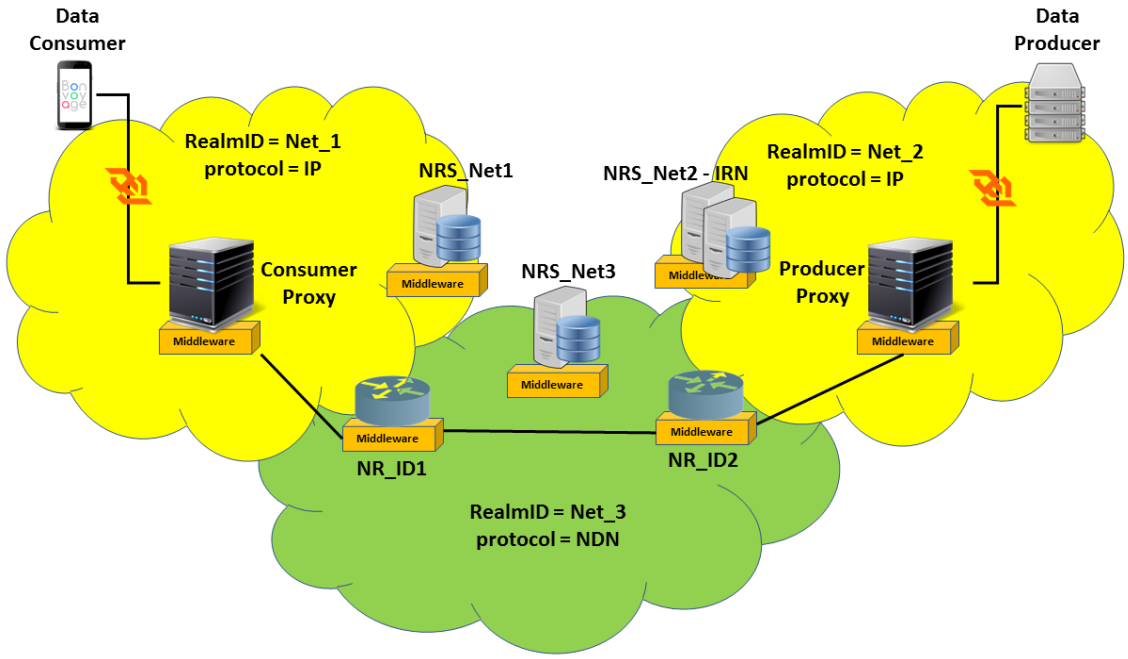

Figure 29: Reference scenario implemented

In the proposed example, it is assumed that a Data Producer generates contents with 1 second pacing under the name/topic *bv/nametest*. These contents include a timestamp and a random number in the range [0-100]. Every time a new content is generated, the couple ⟨timestamp, random_number⟩ is stored within the updated content in a JSON object. The latest 3 generated contents, instead, are always stored within the initial content. To this end, every time a new content is generated, both INIT and UPDATE contents are generated and announced to the system through the front-end APIs **Producer_Publish_Init** and **Producer_Publish_Update**.

Considering the interaction between Name Resolution Services and Named Routers entities, already described in *Deliverable 3.1 - Networking* and recalled in previous Sections, the entire process involves the following steps:

- The Data Producer, implemented as Java Application, initiates a WebSocket connection with the Producer Proxy. After the bootstrap phase, it generates a new content every second under the topic **/nametest** and publishes it in the platform by triggering the **Producer_Publish_Init** and **Producer_Publish_Update** APIs. In both operations, a JSON message is sent. The message contains both the name and the data to be published, formatted according to the scheme described in the previous Section 4. Figure 30 shows the logs reported by the Producer Proxy during the publication of such contents.

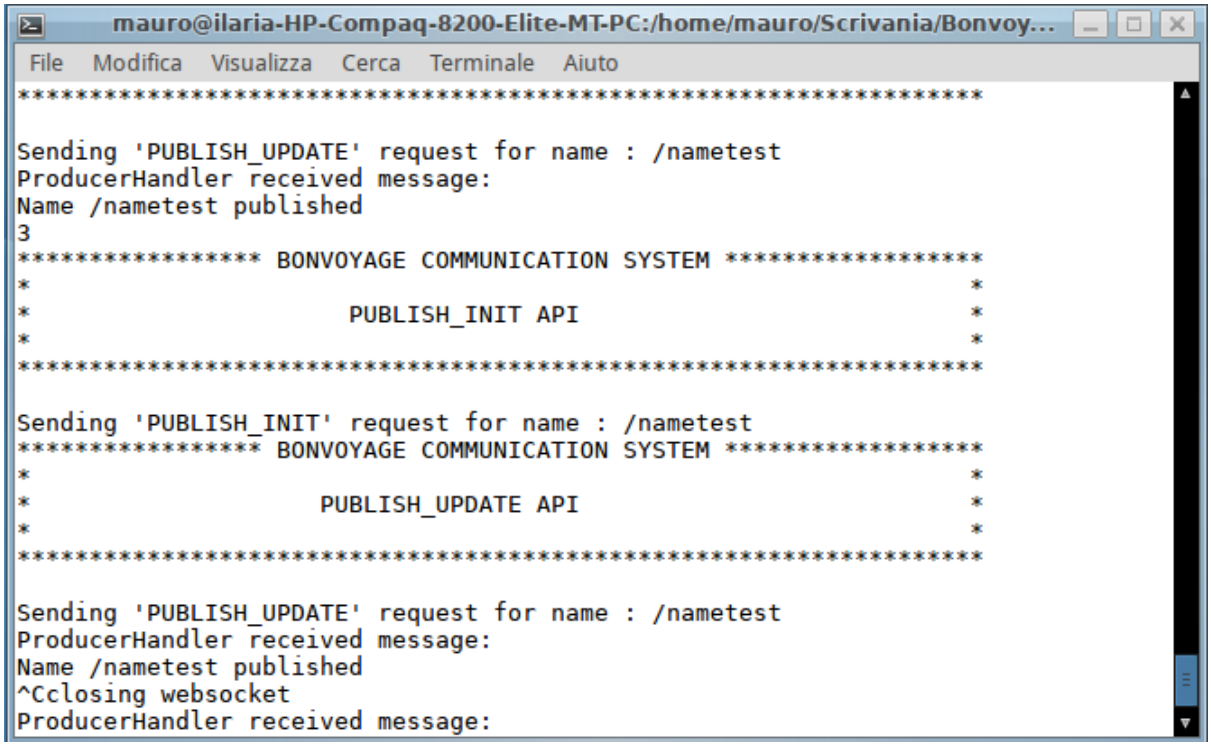

Figure 30: Logs reporting the triggering of front-end APIs by the Data Producer.

- The Producer Proxy receives and processes the publication requests sent by the Data Producer. Thus, it announces the new (or updated) content for the specific topic in the platform. To this end, it uses the **Netw_Announce** to communicate the availability of the new content under the topic **/nametest** to the Internames Rendezvous Node logical node. Figure 31 shows the logs reported by the Producer Proxy during message processing and network operations triggering.

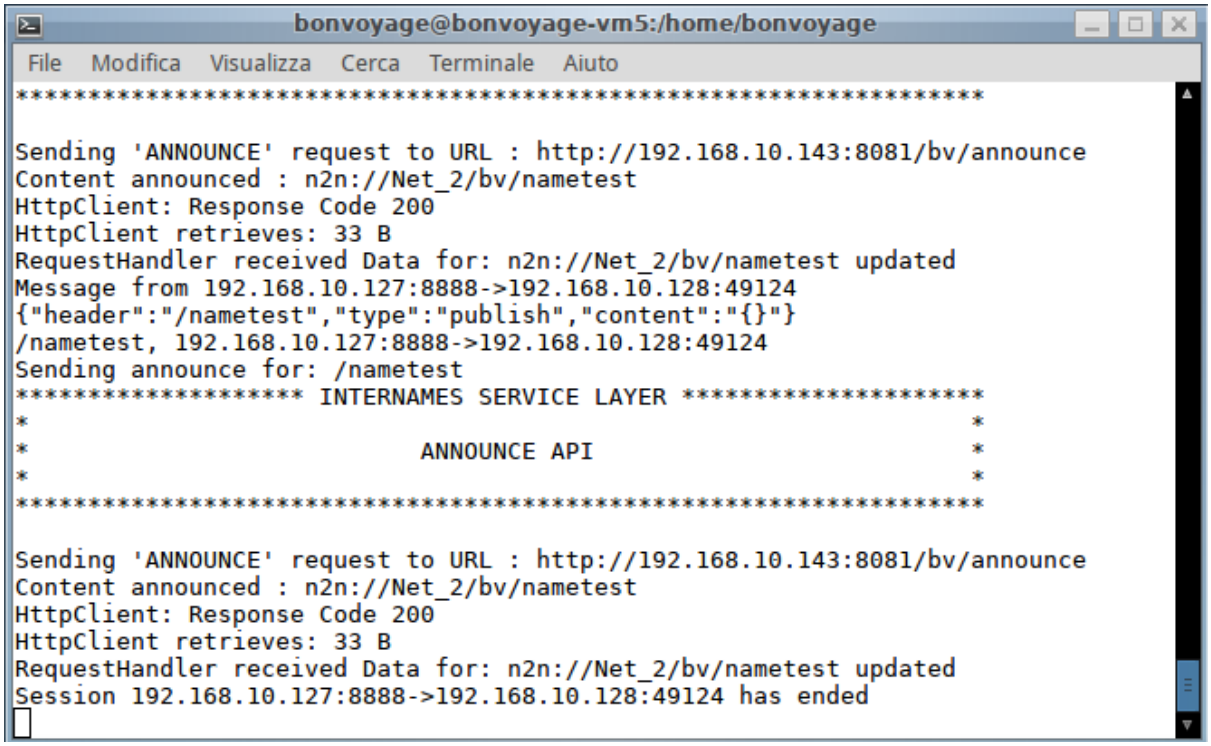

Figure 31: Logs reporting the set of operations executed by the Producer Proxy during the publication process.

Figure 32 shows the logs reported by the IRN logical node during the publish operation.

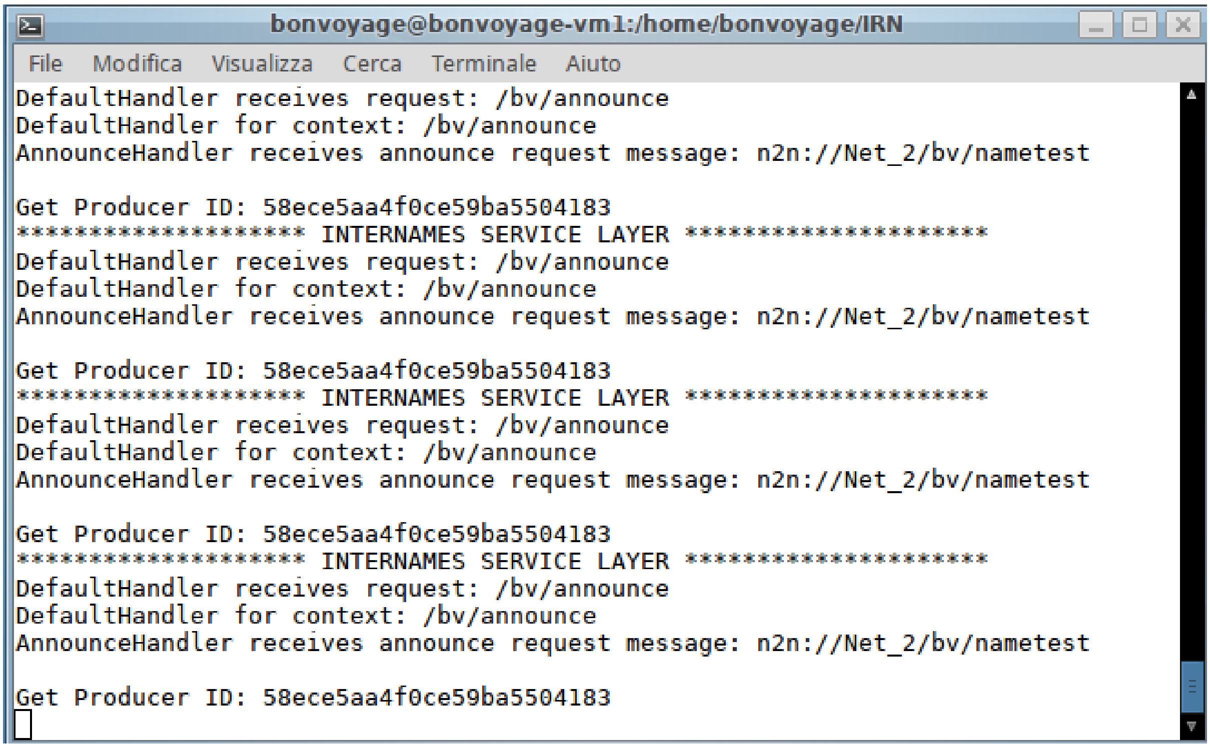

Figure 32: Logs reporting the set of operations executed by the IRN logical node during the publication process.

- The Data Consumer sends the list of names (in this case only the topic **/nametest**) to the Consumer Proxy by using the front-end **Consumer_Subscribe** API, using, as a parameter, the name and the handler dedicated to parsing operations performed for the JSON response. The latter contains the timestamp and the random number;
- The Consumer Proxy receives and processes the requests sent by the Data Consumer, by performing the subscription to the IRN reference node, as shown in Figure 33 and 34;

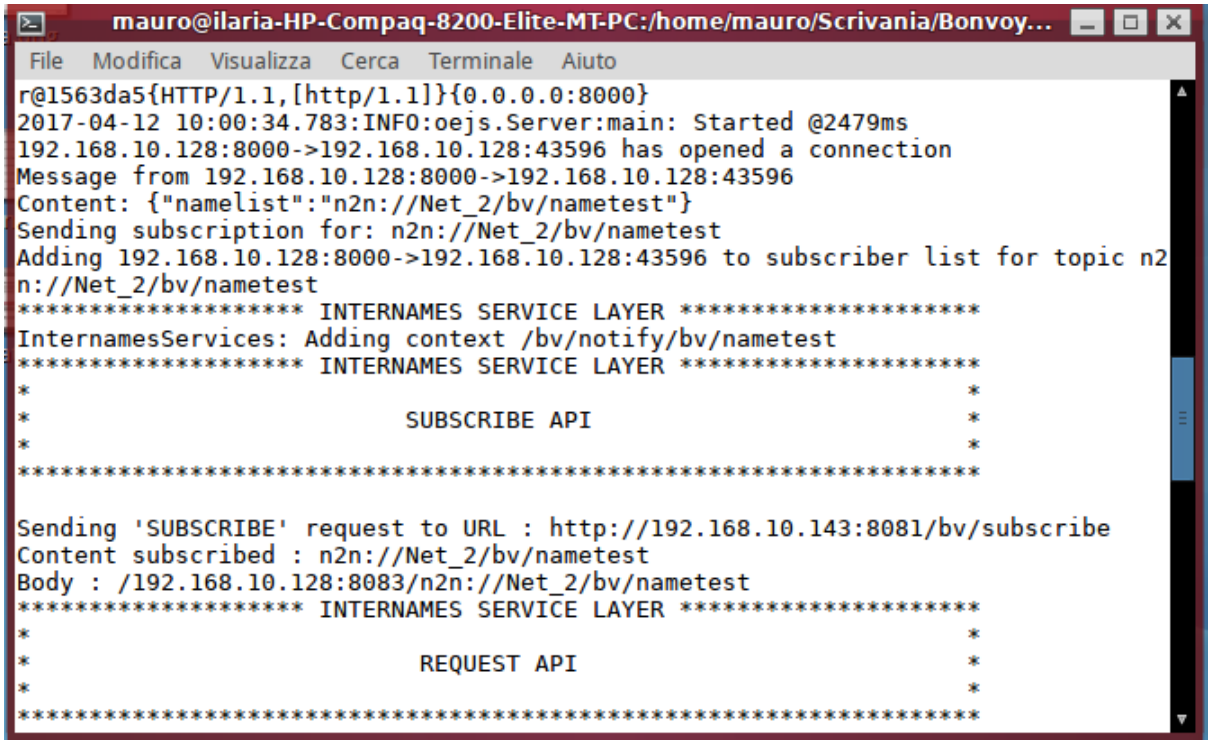

Figure 33: Logs reporting the set of operations executed by the Consumer Proxy during the subscribe process.

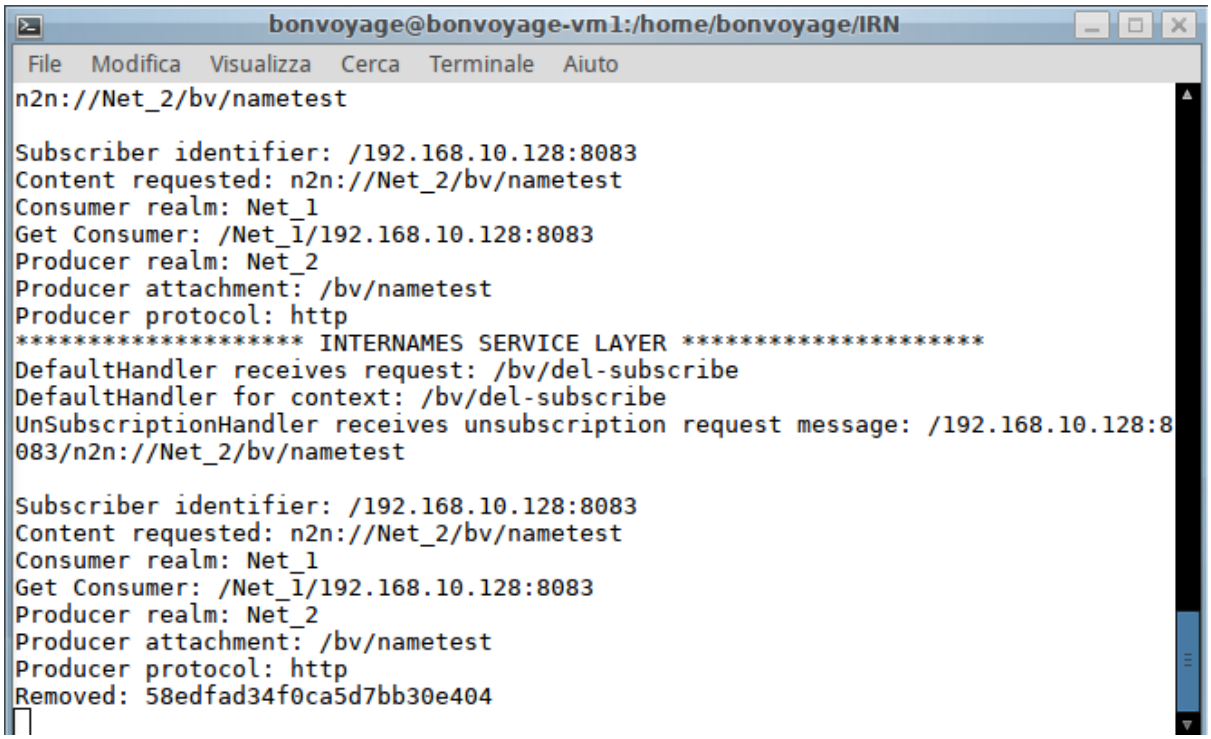

Figure 34: Logs reporting the set of operations executed by the IRN logical node during the subscribe process.

- After perfoming the subscription, the Consumer Proxy must retrieve contents from the BONVOYAGE platform on behalf of the Data Consumer. To this end, it uses the *Netw_-Request* API to retrieve data from the BONVOYAGE Communication System through the Internames Service Layer. This high-level API triggers the networking manager to execute the *low_bv_resolve* and *low_bv_tunnel* atomic operations and the technology manager to generate a GET HTTP message to sent to the reference NRS and NR nodes, respectively. The logs reported during the aforementioned operations are shown in Figure 35.

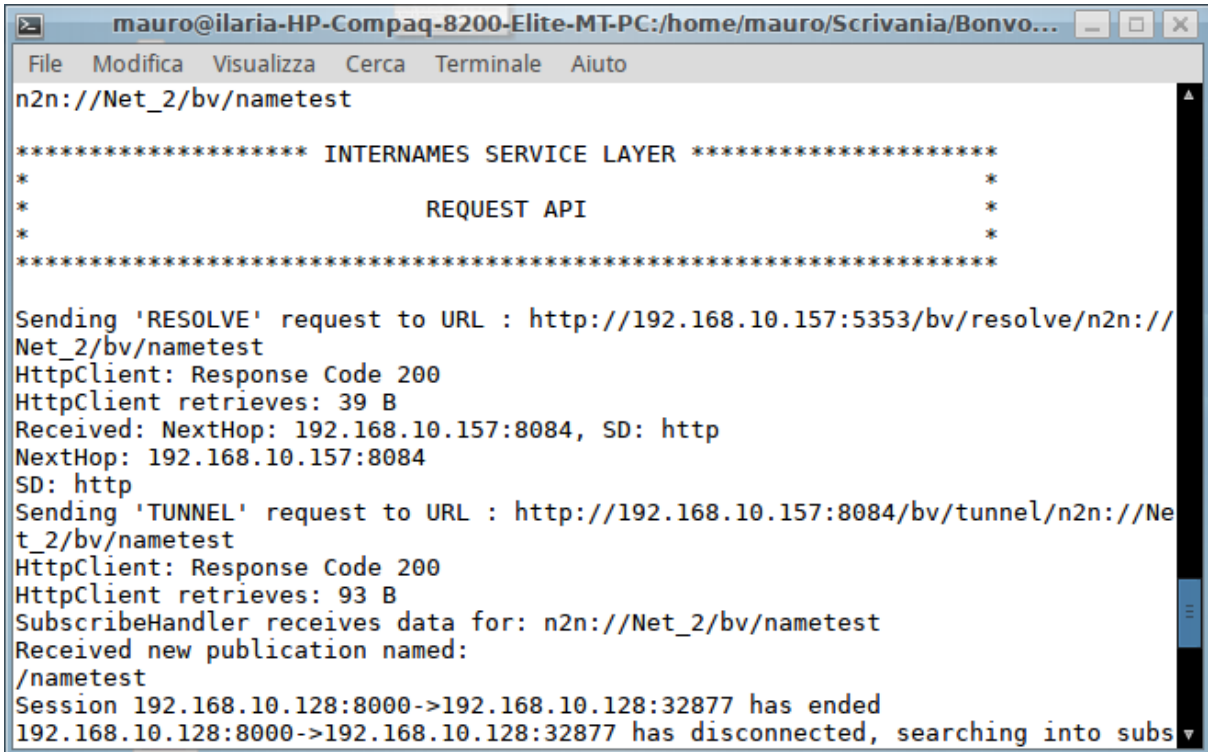

Figure 35: Logs reporting the set of operations executed by the Consumer Proxy during the Request-Response process.

- The *low_bv_resolve* operation, performed by the Consumer Proxy, is translated by the networking manager into an HTTP request to the NRS in the realm *Net_1*, as shown in Figure 36. NRS_Net1 answers by suggesting to send the request towards the Named Router NR_ID_1 using the HTTP protocol. To this end, the *low_bv_tunnel* atomic operation gets involved in the procedure;

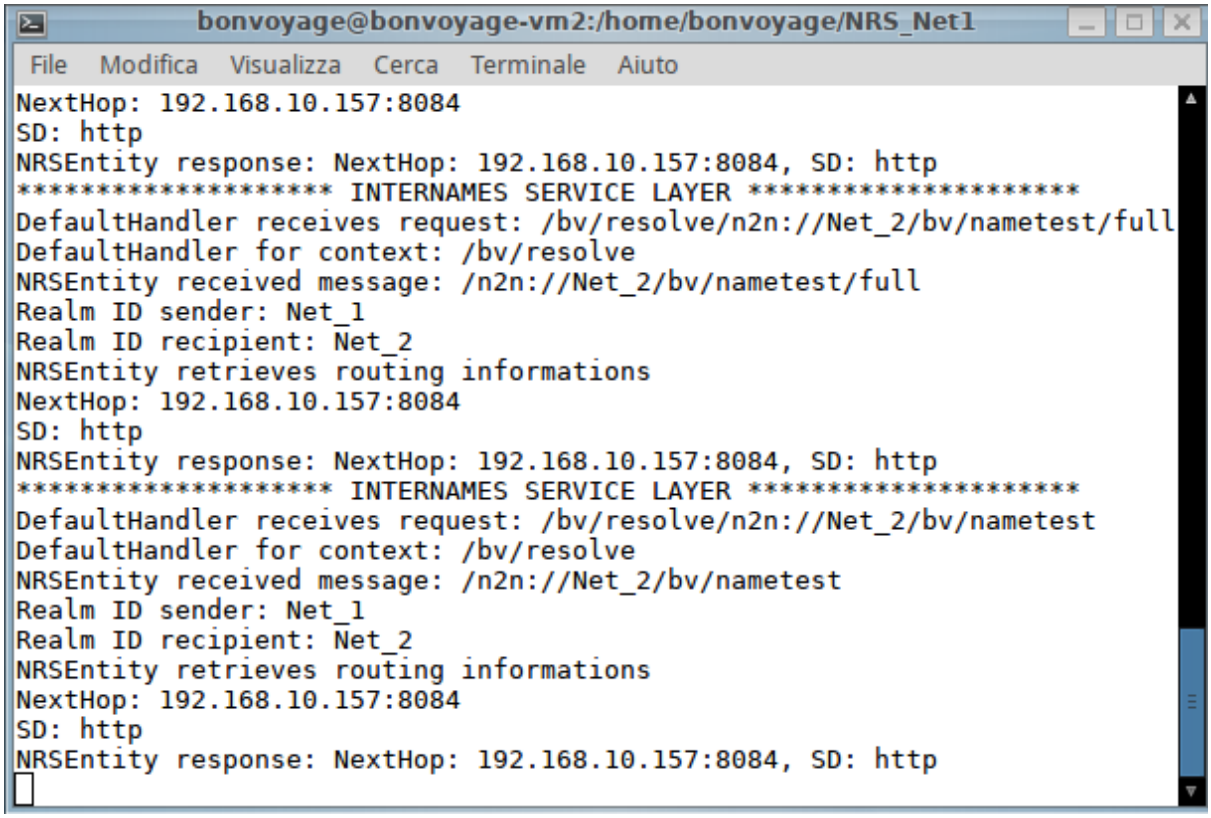

Figure 36: Logs reporting the set of operations executed by NRS_Net1 during the resolution process.

- NR_ID_1 processes the message. As contents have to be retrieved from the BONVOYAGE platform, the NR_ID_1 uses the *Netw_Request* API to retrieve data from the BONVOYAGE Communication System through the Internames Service Layer. This high-level API triggers the networking manager to execute the *low_bv_resolve* atomic operation and the technology manager to generate a GET HTTP message that is sent to the reference NRS node, as shown in Figure 37.

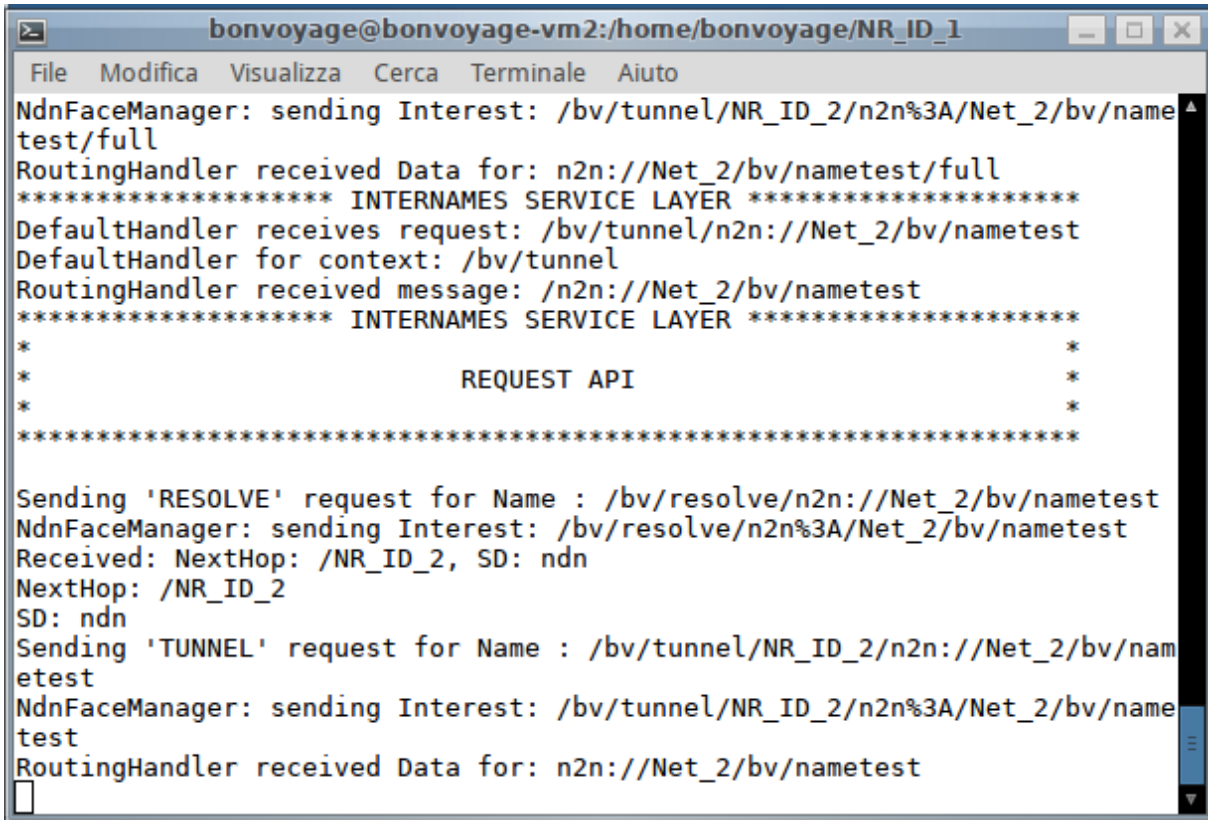

Figure 37: Logs reporting the set of operations executed by NR_ID_1 during the routing process

- NRS_Net3 answers by suggesting to send the request towards the border router NR_ID_2 by using the NDN protocol, as shown in Figure 38. The networking manager of the middleware instance running on the NR_ID_1 executes the *low_bv_tunnel* atomic operation. Therefore, the technology manager sends an NDN INTEREST message to NR_ID_2, asking for the requested contents;

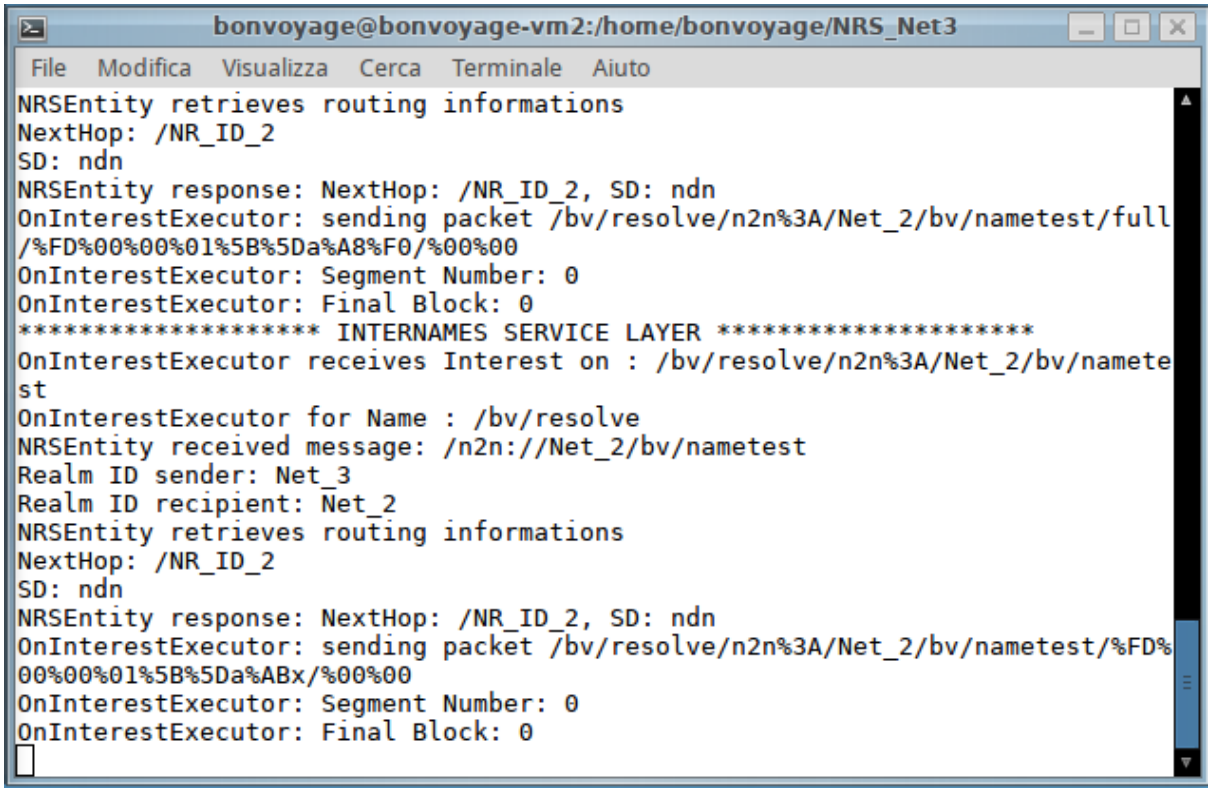

Figure 38: Logs reporting the set of operations executed by NRS_Net3 during the resolution process.

- NR_ID_2 processes the message. As contents have to be retrieved from the BONVOYAGE platform, the NR_ID_2 uses the *Netw_Request* API to retrieve data from the BONVOYAGE Communication System through the Internames Service Layer. This high-level API triggers the networking manager to execute the low_bv_resolve atomic operation and the technology manager to generate a GET HTTP message to sent to the reference NRS node. NRS_Net2 answers by indicating the IP address of the producer proxy that stores the requested data, as shown in Figure 39

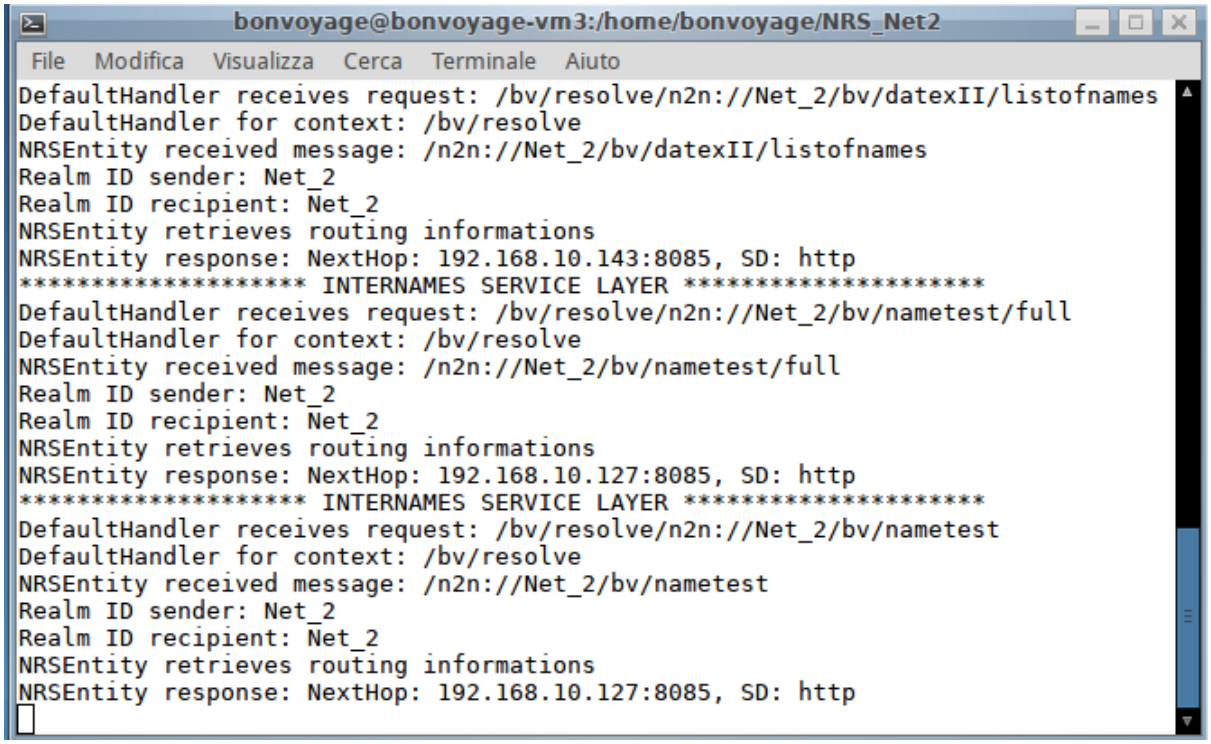

Figure 39: Logs reporting the set of operations executed by NRS_Net2 during the resolution process.

- Once the request is received by the Producer Proxy, as shown in Figure 40, a new data is retrieve from the related Producer through the WebSocket connection. After this, the data is sent back to the Consumer Proxy and then to all the subscribed Data Consumers;

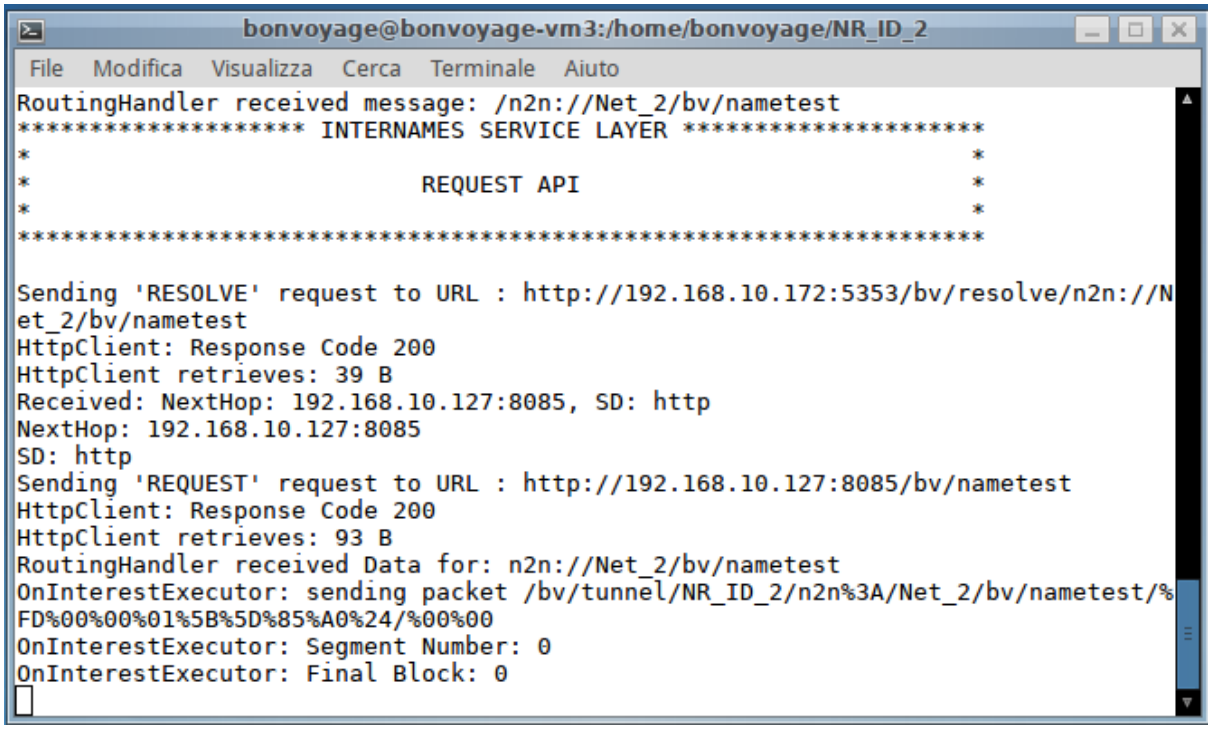

Figure 40: Logs reporting the set of operations executed by NR_ID_2 during the routing process

- The Data Consumer logs the retrieved data, as shown in Figure 41, and stores them according to the types of data received, either is INIT or UPDATE content.

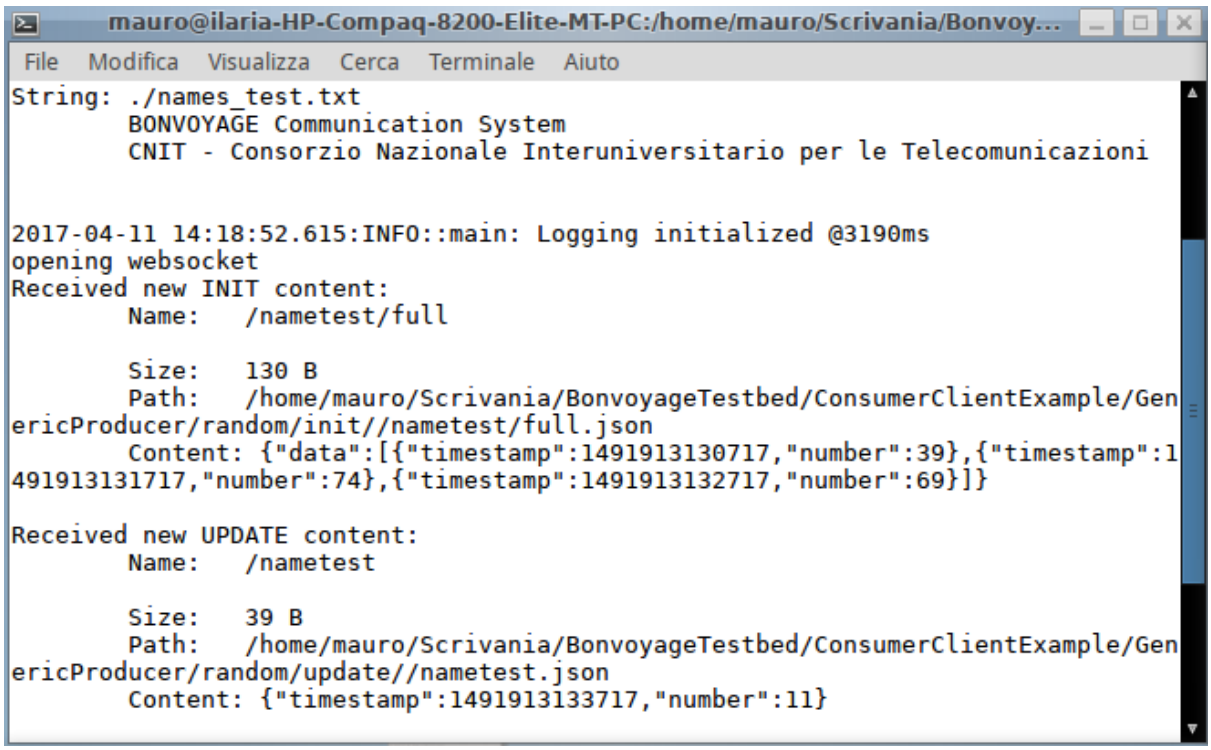

Figure 41: Logs of the Data Consumer, generated when the requested INIT and UPDATE contents are received.

As shown in Figure 42, the INIT file, namely */nametest/init.json*, contains the three latest entries generated since the Data Consumer has performed the subscription, and the UPDATE file, namely */nametest.json*, containing the latest punctual update.

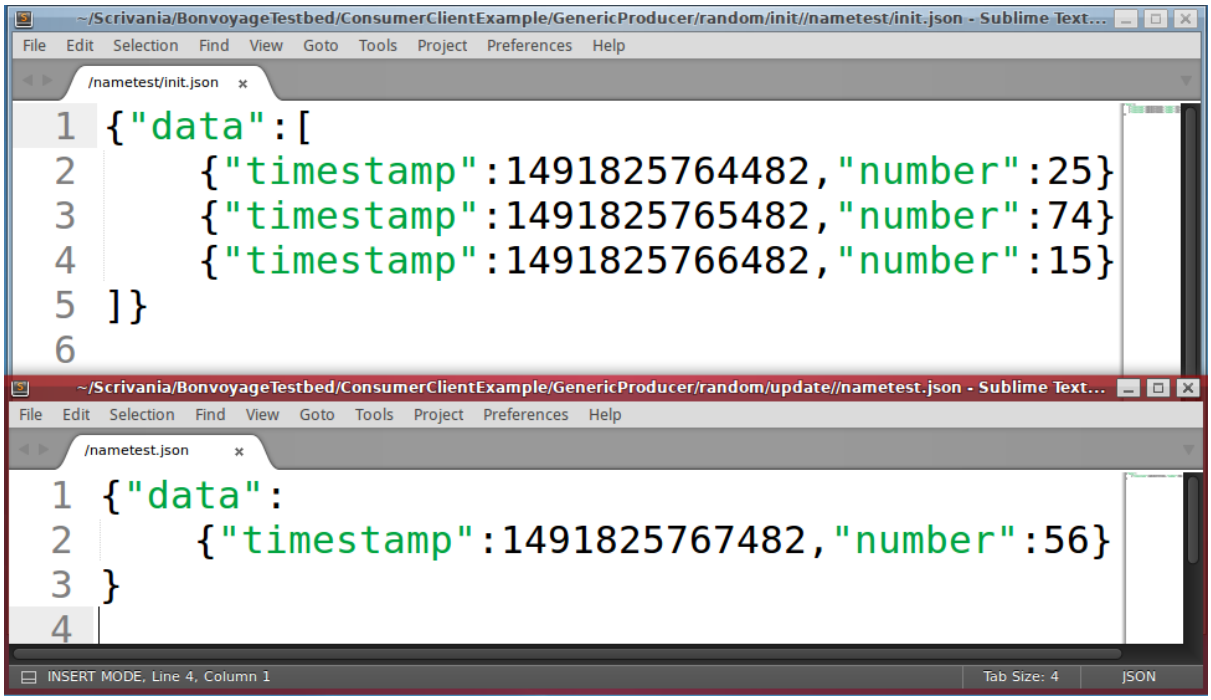

Figure 42: INIT and UPDATE files stored by the Data Consumer

As a final remark, the developed testbeds demonstrated the functionalities offered by the BONVOYAGE Communication System, which exploits both Request-Response and Publish-Subscribe schema. The **Data Producer** is able to publish the generated data in BONVOYAGE platform, managing different content types according to its customized logic. On the other side, the **Data Consumer** is able to collect data generated in BONVOYAGE platform and receive the related updates, if any. Both operations are executed 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.

References

[1] 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.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," in *ACM SIGCOMM Computer Communication Review (CCR)*, Jul. 2014.

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

[4] E. Commission, "Directive 2010/40/eu of the european parliament and of the council of 7 july 2010 on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport," 2010.

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

[6] Y. Zhang, D. Raychadhuri, L. A. Grieco, E. Baccelli, J. Burke, R. Ravindran, G. Wang, A. Lindren, B. Ahlgren, and O. Schelen, "Requirements and Challenges for IoT over ICN, IRTF Internet Draft, draft-zhang-icnrg-icniot-requirements-00," IRTF, Internet Draft, Nov. 2015.

[7] 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.

[8] 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.

[9] A. Detti, N. B. Melazzi, M. Orru, R. Paolillo, and G. Rossi, "Opengeobase: Information centric networking meets spatial database applications," *2016 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, Dec 2016.

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

[11] Y. Zhang, D. Raychadhuri, L. A. Grieco, E. Baccelli, J. Burke, R. Ravindran, G. Wang, A. Lindren, B. Ahlgren, and O. Schelen, "Design Considerations for Applying ICN to IoT," Internet Engineering Task Force, Internet-Draft, March 2017.

[12] "OpenMote," 2016. [Online]. Available: http://www.openmote.com/

[13] Thubert, P. and Watteyne, T. and Assimiti, R. A. (2014), "An architecture for IPv6 over the TSCH mode of IEEE 802.15.4 draft-IETF-6TiSCH-architecture-11," 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-architecture-11

[14] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "Openwsn: A standards-based low-power wireless development environment," *Wiley's Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[15] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Communications*, vol. 21, pp. 20–26, 2014.

[16] S. K. Datta and C. Bonnet, "Smart m2m gateway based architecture for m2m device and endpoint management," *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE*, 2014.

[17] M. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "Om2m: extensible etsi-compliant m2m service platform with self-configuration capability," *Procedia Comput. Sci.*, vol. 32, pp. 1079–1086, 2014.

[18] "Pandaboard," 2016. [Online]. Available: http://pandaboard.org/