TECHNISCHE UNIVERSITÄT BERLIN

Unification of Monitoring Interfaces of Federated Cloud and Future Internet Testbed Infrastructures

- Engineering Doctorate Dissertation -

Yahya Al-Hazmi, **M.Sc.**

Berlin, 5. January 2016
D 83

Unification of Monitoring Interfaces of Federated Cloud and Future Internet Testbed Infrastructures

vorgelegt von
Yahya Al-Hazmi, **M.Sc.**

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Sebastian Möller
Gutachter:	Prof. Dr.-Ing. Thomas Magedanz
Gutachterin:	Prof. Dr.-Ing. Ina Schieferdecker
Gutachter:	Prof. Dr. Serge Fdida

Tag der wissenschaftlichen Aussprache: 5. January 2016

Berlin, 2016
D 83

Abstract

The federation of information and communication technology infrastructures is gaining significant attention from academia and industry, in particular in the fields of cloud and Future Internet testbeds, where heterogeneous resources and services are shared across multiple administrative domains. This is because of its multilateral benefits for users and infrastructure providers in terms of increasing the capacity of resources and diversity of offerings, as well as efficient resource utilization and complementarity.

A major challenge in such federated environments is the exchange of monitoring data across the federation due to the variety and heterogeneity of the tools and interfaces used. Much effort has been made to unify interfaces by the use of common data models and protocols that are task or domain specific, particularly in the network domain. This hinders their application in other domains without significant modifications, becoming even more complex in federated, heterogeneous domains.

This thesis addresses this issue and provides an architecture for the unification of monitoring interfaces to enable a unified data representation across federated, heterogeneous infrastructures. This is achieved by three mechanisms: first, an adaptation layer on top of the tools deployed at infrastructure level is responsible for providing the data in a unified representation through a common interface; second, the use of a flexible, extensible and schema-independent data transportation protocol; and third, the use of a common information model that is based on Semantic Web technologies and caters for the collection and representation of data in unified, meaningful manner, as well as facilitating interoperability between tools and data consolidation.

This thesis delivers three major contributions as the main research results. These are an extensible monitoring architecture for federated, heterogeneous infrastructures, an ontology-based information model for describing monitoring related concepts and relations in federated, heterogeneous infrastructures at the conceptual and semantic levels, and a prototype implementation that has been validated through selected use-case projects.

Zusammenfassung

Die Föderation von Informations- und Kommunikationstechnologie-Infrastrukturen erhält zur Zeit große Aufmerksamkeit der Forschung und der Industrie. Insbesondere in den Bereichen Cloud und Future Internet Testumgebungen, wo heterogene Ressourcen und Dienstleistungen über die Grenzen administrativer Domänen hinweg bereitgestellt und gemeinsam genutzt sind, zeigen sich die vielseitigen Vorteile für Nutzer und Infrastrukturanbieter im Hinblick auf die Erhöhung der Kapazität der Ressourcen und Angebotsvielfalt sowie effiziente Ressourcennutzung und Komplementarität.

Aufgrund der Vielfalt und Heterogenität der verwendeten Werkzeugen und Schnittstellen jedoch, ist der föderationsweite Austausch von Monitoringdaten eine große Herausforderung in solchen föderierten Umgebungen. Viele Ansätze versuchen Schnittstellen durch die Verwendung gemeinsamer Datenmodellen und Protokollen zu vereinheitlichen, doch sind diese aufgaben- oder domänenspezifisch, insbesondere in der Netzwerkdomäne. Dies behindert ihre Anwendung in anderen Domännen ohne nennenswerte Modifikationen, besonders in den komplexen föderierten, heterogenen Domännen.

Die vorliegende Arbeit befasst sich mit diesem Thema und bietet eine Architektur für die Vereinheitlichung der Monitoringschnittstellen, um eine einheitliche Darstellung von Daten innerhalb von föderierten, heterogenen Infrastrukturen zu ermöglichen. Dies wird durch drei Mechanismen erreicht. Zunächst ist eine Anpassungsschicht auf der eingesetzten Werkzeuge auf Infrastrukturebene für die Bereitstellung der Daten in einer einheitlichen Vertretung über eine gemeinsame Schnittstelle zuständig. Der zweite Mechanismus ist die Verwendung eines flexiblen, erweiterbaren und schemaunabhängigen Datentransportprotokolls. Schließlich ist die Verwendung eines gemeinsamen Informationsmodells, das auf Semantic Web Technologien basiert und für die Erfassung und Darstellung von Daten in einer einheitlichen, sinnvollen Art und Weise, sowie die Erleichterung der Werkzeuginteroperabilität und Datenkonsolidierung sorgt, der dritte Mechanismus.

Diese Arbeit liefert drei Hauptbeiträge als Forschungsergebnisse. Diese sind eine erweiterbare Monitoringarchitektur für föderierte heterogene Infrastrukturen, ein Ontologie-basiertes Informationsmodell für die Beschreibung von Monitoring-bezogene Konzepten und Relationen in föderierten heterogenen Infrastrukturen auf der konzeptionellen und semantischen Ebenen, und eine Prototyp-Implementierung, die durch ausgewählte Use-Case-Projekte validiert wurde.

Acknowledgments

For making this research and thesis possible, I owe a debt of gratitude to a number of people and institutions. This work would not have been possible without their help and support in different ways.

First, I would like to thank Prof. Dr.-Ing. Thomas Magedanz for his support and guidance throughout my research work. I would also like to thank Prof. Dr.-Ing. Ina Schieferdecker and Prof. Dr. Serge Fdida for their helpful support and valuable suggestions.

Second, I would like to express my sincere thanks to everyone who assisted me directly or indirectly, professionally or personally to complete this work. Special thanks go to my colleagues at Technische Universität Berlin AV and Fraunhofer Institute FOKUS NGNI. I'm very grateful to Andisa Dewi, Robyn Loughnane, Mingyuan Wu, Abdulrahman Hamood, Alexander Willner, Adel Al-Hezmi and Daniel Nehls for their support and help.

I deeply thank my wife Maha for her love, time and understanding. She was always behind me and gave her unconditional support, even when that meant sacrificing the time we spent together.

I would like to thank my family for their wonderful support. I owe special thanks to my mother Mohelah and my son Aiman.

Finally, I would like to thank the European Commission for funding the FIRE and FI-PPP initiatives, and the teams I worked with within these initiatives for their valuable discussions and feedback.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

1.1. Context and Motivation

THE Internet has transformed our lives in many domains – social, economic, educational, personal, etc. It has become therefore an integral part of our daily lives. Over the past years, the Internet has evolved quickly and become a major tool for many services in various fields such as eHealth, public services, job search, tourism, entertainment, and more. Thus, the number of users and devices connected to the Internet is growing dramatically as well as the amount of Internet data traffic increases rapidly [1]–[3]. This growth in both number of devices and the amount of data will further increase through new technologies and communication paradigms like Machine-To-Machine Communication (M2M) [2]. In parallel, a plethora of applications and services are being introduced that require fast, reliable and secure transmission of data, such as video, 3D content, online banking, etc.

Management of the continuously growing Internet and networks is becoming more complex. Furthermore, the phenomenal success of the Internet is a catalyst for new and innovative applications, products and ideas, creating expectations beyond its current capability to support the demands of such applications and services to a sufficient level [4], [5]. This is because the Internet was not designed for the current

level of usage. New technologies and future architectures [6] are being introduced to cope with the limitation of the current Internet architecture and its management complexity in terms of improving the scalability, Quality of Service (QoS), resource utilization and efficiency, reliability, and security.

In addition, new communication and networking paradigms and technologies – such as Cloud Computing, Software Defined Networking [215] (SDN), Network Function Virtualization [193] (NFV), M2M and Internet of Things [189] (IoT) – are emerging and changing the current Information Technology (IT) and Telecommunication worlds. On the one hand, they bring forth new business models and opportunities to build Smart Cities, eHealth, eGovernments, eLiving, and other future services and applications. However, on the other hand, they add more management complexity.

Important to the development of the Future Internet (FI) is how these new technologies, architectures and their associated applications are trialed and evaluated. There are currently a large number of activities worldwide focusing on building testbeds for experimenting and prototyping Smart Cities and FI architectures, services and applications [6]. A testbed as defined in [7] is as follows.

Definition of Testbed: "An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test".

However, for the scope of this thesis the definition will be limited to Information and Communication Technology (ICT) environments. In this thesis, the term 'experimentation facility' is also used as a synonym for 'testbed'. Major examples of relevant testbeds include the Global Environment for Network Innovations[1] [13], [188] (GENI) and PlanetLab [208], [209][2] (PlanetLab) in the United States (US); the Future Internet Research and Experimentation[3] [65] (FIRE) and Future Internet Public Private Partnership[4] [181] (FI-PPP) testbeds in Europe; and the German Lab[5] [187] (G-Lab) in Germany. They each have different infrastructures based on various technologies and offer services for diverse experimentation communities.

Collaboration among multiple distributed testbeds to provide a larger pool of shared heterogeneous resources is recognized to significantly enrich and accelerate FI research and development. Indeed, sharing and granting access to resources beyond the boundaries of independent administrative domains has multifold benefits: i) complementarity of resources in the field of service-composition (e.g. Internet and Telecom), ii) increased resource utilization, and iii) increased return on infrastructure investments. This is referred to as *resource federation* in the literature [8]. This

[1] http://geni.net
[2] http://planet-lab.org
[3] http://ict-fire.eu
[4] http://fi-ppp.eu
[5] http://german-lab.de

concept is applied in the area of distributed and grid computing, and is becoming a hot topic in Cloud Computing and FI testbed federation. The most proper definition of a federation in the context of this thesis is given in [9] as follows.

Definition of Federation: "A model for the establishment of a large scale and diverse infrastructure for the communication technologies, services, and applications and can generally be seen as an interconnection of two or more independent administrative domains for the creation of a richer environment and for the increased multilateral benefits of the users of the individual domains".

There is ongoing work on FI testbed federation to deliver large-scale and higher performance experimentation facilities, e.g. the efforts within the context of FIRE [10]–[12] and GENI [13], [14] initiatives. In the field of Cloud Computing, federation is recognized as being of tremendous value to the cloud industry, with the increasing number of commercial cloud providers (in particular the Small and Medium Enterprises (SMEs)) and their broad diversity of offerings, as well as the numerous end consumers [15]. Many research activities focus on cloud federation and interoperability, with examples including the Cross-Cloud Federation Manager [16], BonFIRE Multi-Cloud Test Facility [17], IEEE Intercloud [18], Future Internet Core Platform[6] [184] (FIWARE) Lab Cloud Federation [19], and the European Grid Infrastructure (EGI) Federated Cloud Task Force.[7]

Nevertheless, federation is still evolving and is recognized as a potential field of research associated with several challenges [20]. Two of the most critical challenges in realizing a federation on much larger scales are the monitoring and control of heterogeneous resources across multiple administrative domains [21], [22]. Powerful and convenient tools offered through common Application Programming Interfaces (APIs) are required to support monitoring and controlling the involved infrastructures' resources and services. These APIs enable their interoperability, and provide offerings to the different user communities involved in a common and standardized manner.

Generally, monitoring is essential in a number of areas and plays a significant role in the management and control of such complex, evolving ecosystems. Besides what Tom DeMarco said namely "you cannot control what you cannot measure" [23], measurement and monitoring are fundamental pillars of experimental-based scientific research, where emerging, advanced computing and networking technologies are tested and evaluated. To this end, this thesis focuses on monitoring services (including measurement) in federated infrastructures, taking Cloud Computing and FI testbeds as two practical use cases.

Advanced monitoring services are required to fulfill the demands of i) the emerging technologies and communication paradigms (e.g. monitoring capabilities to support

[6] http://fi-ware.org
[7] https://www.egi.eu/infrastructure/cloud/

the key characteristics of Cloud Computing, like multi-tenancy and high availability [24]), and ii) the federation of independently administered infrastructures (e.g. monitoring capabilities to support key features and aspects of the federation, such as the scalability, extensibility, user-centricity "decoupling services from infrastructures", interoperability, trustworthiness and Service Level Agreement (SLA) management).

There are many definitions for monitoring. However, the most appropriate definition is the one defined by the ITIL Service Management [25] as follows: "repeated observation of a configuration item, IT service or process to detect events and to ensure that the current status is known". In line with this, the monitoring definition within the context of this thesis is as follows:

Definition of Monitoring: The process of constantly observing and recording information about resources (virtual and physical devices, systems, processes, applications, networks, traffic flow, etc.) to determine their state, usage, performance, and behavior. This information is used by various users and systems that are responsible for controlling and managing these resources, as well as further services such as capacity planning, SLA management, trustworthiness, security and privacy assurance, data analytics, etc.

Figure 1.1 summarizes the main motivations for the current research work.

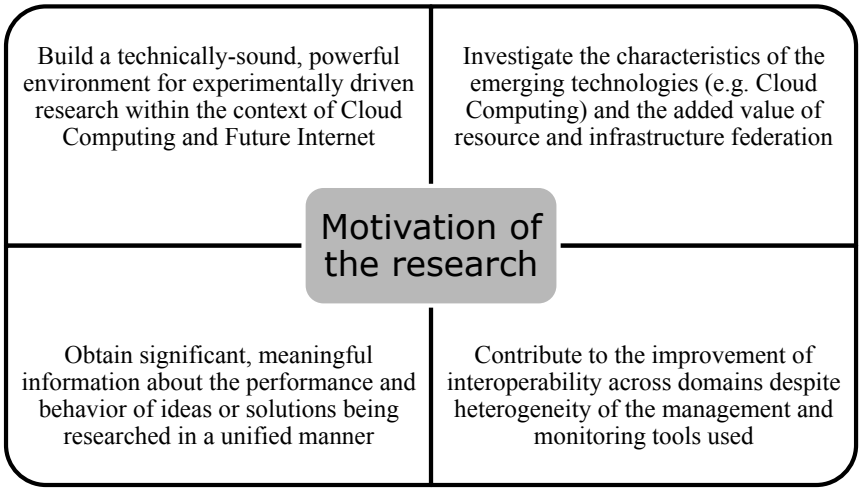


Fig. 1.1.: Main motivations of the thesis

1.2. Problem Statement and Formulation

There are a large and diverse number of measurement and monitoring tools and frameworks addressed in the literature in different contexts. They differ from each other in their architectures, functionalities, usage, and where measurements take

place. Some tools are used for system monitoring (e.g. Central Processing Unit (CPU) usage, memory consumption), while others used for monitoring network performance (e.g. packet loss, delay and throughput) or applications. Tools might be, on the one hand, limited to execute only particular measurements (active or passive), status information, notifications and logs. On the other hand, some tools could partially support cross-layer monitoring, expanding from infrastructure resources through the network up to services and applications.

For clarification and to define the scope of this thesis, Figure 1.2 illustrates a taxonomy of monitoring tools. The taxonomy has been built inspired by the one provided in [26] that is extended to cover more aspects, which will be discussed throughout this thesis.

The taxonomy is built according to the fact that the majority of the tools have features in common. They are organized into seven categories: architecture, communication paradigm, core functionalities, data collection and transportation, data representation, data access, and primary use cases. Each of these categories includes various sub-classifications. To be noticed that only some sub-classifications are represented as examples besides those considered in this thesis that are shaded in gray. Indeed, some tools fall under numerous classifications, whereas others are restricted to a single category. An example of the former is the Zabbix[8] monitoring tool that has a distributed architecture, client-server communication paradigm, includes data producers, collectors and viewers. Zabbix provides the data in push and pull modes in structured representation that can be then access either through a Graphical User Interface (GUI) or JavaScript Object Notation - Remote Procedure Call (JSON-RPC) API. Zabbix can be applied in multiple areas such as health and status, infrastructure resources, and network monitoring. The Ping program (Internet Control Message Protocol (ICMP) echo) is an example of the latter case, where it can only be classified under the type "producer" as a probe for network measurements (i.e. its application area) and the data is represented in a structured manner.

Moreover, in federated environments where infrastructures are administered separately, from different domains (cloud-based, OpenFlow-enabled, WiFi, sensor networks, mobile networks, etc.) providing heterogeneous resources, the heterogeneity and variety of the monitoring solutions increase. Furthermore, an infrastructure might even use several tools in order to monitor the entire facility or to provide cross-layer monitoring. To give an example from one domain, namely Cloud Computing, two individual surveys [26], [27] on cloud monitoring tools (considering their capabilities and limitations) have shown how the state-of-the-art tools are not capable of fulfilling all monitoring requirements. In their surveys, the authors did not consider aspects relevant to federation that would have raised additional requirements, and thus, limit the capabilities of these tools even more.

[8] http://www.zabbix.com

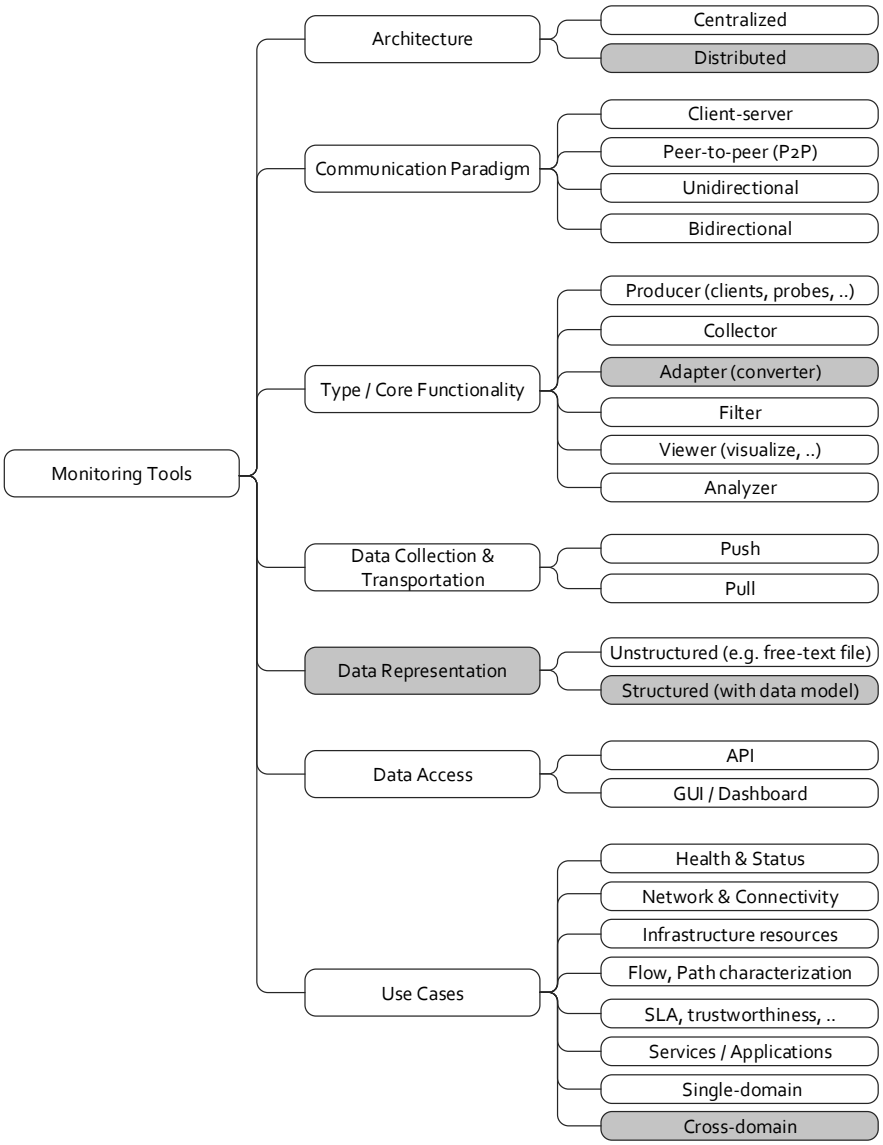


Fig. 1.2.: Monitoring tool taxonomy

State-of-the-art monitoring solutions mainly target homogeneous, single-entity-administered infrastructures. There is no existing solution that is capable of monitoring complex, large-scale federated environments, as well as supporting the various monitoring services and capabilities required by diverse users. Users can be administrators and operators, end-users (experimenters), as well as a couple of management systems responsible for ensuring availability and performance of the offered resources and services, interoperability, SLA conformance, trustworthiness, privacy, etc.

There is therefore a need for a monitoring solution that operates across a federation, aggregating a multitude of measurements from various sources of different infrastructures, and provides various sets of data for many consumers (in both central and

distributed modes) in a unified and standardized manner. The significant need for such a monitoring solution has been identified in a couple of research projects that focus on the federation of heterogeneous FI testbeds [11] and on cloud federation [17], [28], as well as in research calls [29], [30], and in a survey on cloud monitoring [31].

The proposed solution should accommodate a wide range of measurement and monitoring services at different levels of granularity. This covers providing cross-layer monitoring information from low-level infrastructure resources up to services and applications specific measurements for various groups of users who have different needs and interests. Usually, to provide cross-layer monitoring information in a single infrastructure, various tools are used depending on where the measurements take place, as traditional tools focus on monitoring homogeneous resources (i.e. a single layer) [32]. They are used to monitor infrastructure resources (physical and virtual), network connectivity and performance, and running services and applications. The heterogeneity of the tools used increases even more in a federated environment. From this perspective, independently administered infrastructures might use heterogeneous tools that use different data formats, data models and schemas, APIs, and databases. This results in providing the collected monitoring and measurement data across the federated infrastructures in different formats and structures, and leads to disparate interpretation and semantic interoperability problems.

This problem is well known in the field of heterogeneous database systems, and there are methods for their integration and interoperability in the literature [33]. Although this is out of the scope of this thesis and its requirements, some knowledge from this domain can be applied to this work. As long as there is disagreement on meanings, interpretations or intended use of information, semantic ambiguity can arise [33]. Examples of possible heterogeneities include the following:

- *Naming conflicts*: Tools could use different names to represent the same concept. For example, the number of CPUs can be named by tool A as *num_ cpus*, while in tool B *cpu_ count*.

- *Unit conflicts*: Tools could use different values to represent the same concept. For example, tool A provides the amount of the free memory in kilobytes while tool B in bytes.

- *Different data structures or schemas*: Tool A provides only the name of the measured metric with its measured values, while tool B provides the name, measures, units, and further information.

- *Different programmatic access*: Tool A provides the data through a JSON-RPC API while tool B via a direct access to its database (e.g. MySQL) or through a GUI frontend.

- *Different implementations*: Even if tools expose the same type of APIs, they might have different implementations and thus different names. Similarly, tools that use the same databases might use various schemas.

At the beginning of the research work for this thesis there were already some efforts [34]–[36] focused on developing information models to enable data to be collected from different tools in a unified format, but these were limited to the needs of the problems in question (domain and task specific). The heterogeneity problem still remains in the case of federated infrastructures comprising heterogeneous domains, as well as diverse tasks within each domain.

To overcome this problem that was addressed by several major federation projects [11], [19], [28], [34], the target monitoring solution must allow monitoring data to be aggregated from different sources across the federated infrastructures (that maintain the use of tools already in place) and provide the data in a uniform format. This will eliminate misinterpretation of data by its users, as well as facilitate the interoperability of infrastructures in terms of monitoring data exchange.

Research aimed at achieving such a monitoring solution has become even more prevalent and critical due to the demand of emerging technologies (Clouds, IoT, Big Data, etc.) and cooperation strategies (e.g. infrastructure federation). With this in mind, the research literature still lacks a valid, generic solution. However, there are some efforts in the literature to partially solve the problem, by proposing solutions for individual sub-problems: i) cross-layer monitoring [32], ii) inter-domain connectivity monitoring [37], [38], and iii) unification of data but limited to specific domains [34]–[36]. These are discussed in detail in Chapter 2 on the state-of-the-art.

Figure 1.3 summerizes the main incentives for the research work conducted in this thesis.

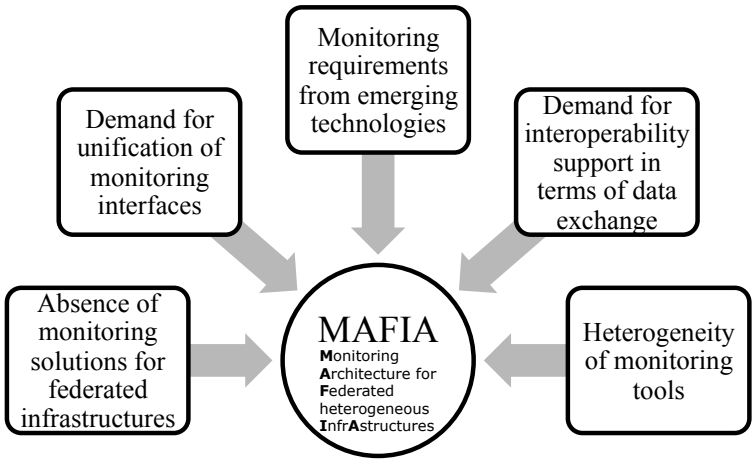


Fig. 1.3.: The incentives of the research work

1.3. Objectives and Research Questions

The objective of this thesis is to design and develop a monitoring solution operating across federated infrastructures. It has to be capable of unification of monitoring interfaces, which already takes place at the individual infrastructures exposing data in different formats. Unification is achieved through a common interface that allows exposing the data in a uniform format for multiple stakeholders who have various needs.

The major contribution of this thesis is to answer the following research questions derived from the problem stated in Section 1.2 within the context of federated clouds and FI testbeds.

Q1: *How to design an architecture that i) allows the integration and management of heterogeneous monitoring solutions distributed in a federated environment in order to provide a set of monitoring services in a common manner, and ii) is extensible to allow monitoring of other similar fields of application?*

A number of considerations are to be taken into account when designing such an architecture, e.g. infrastructures of different natures, monitoring systems with limited access possibilities, etc.

Q2: *How to model heterogeneous monitoring and measurement related concepts and relationships allowing the target solution to provide the data in a common and meaningful way?*

While working on the model design of the target model, standardization should be kept in consideration throughout the whole design process.

1.4. Scope of the Thesis and Major Contribution

This section positions the scope of the thesis within the context of Cloud Computing and FI research and experimentation.

The thesis is built on current research in infrastructure federation that is recognized to be at the core of cloud platforms, future Internet and networks. Although the work has a strong focus on these domains, the concepts are general enough to be applicable to other similar domains as well.

Federation of ICT infrastructures is a non-trivial task and gives arise to several technical and research challenges [10], [14]. Figure 1.4 presents the major aspects or challenges of a federation. It is an extended version of the one introduced in [39]. Amongst these aspects, the focus of this thesis lies on the monitoring, as shaded in gray in Figure 1.4.

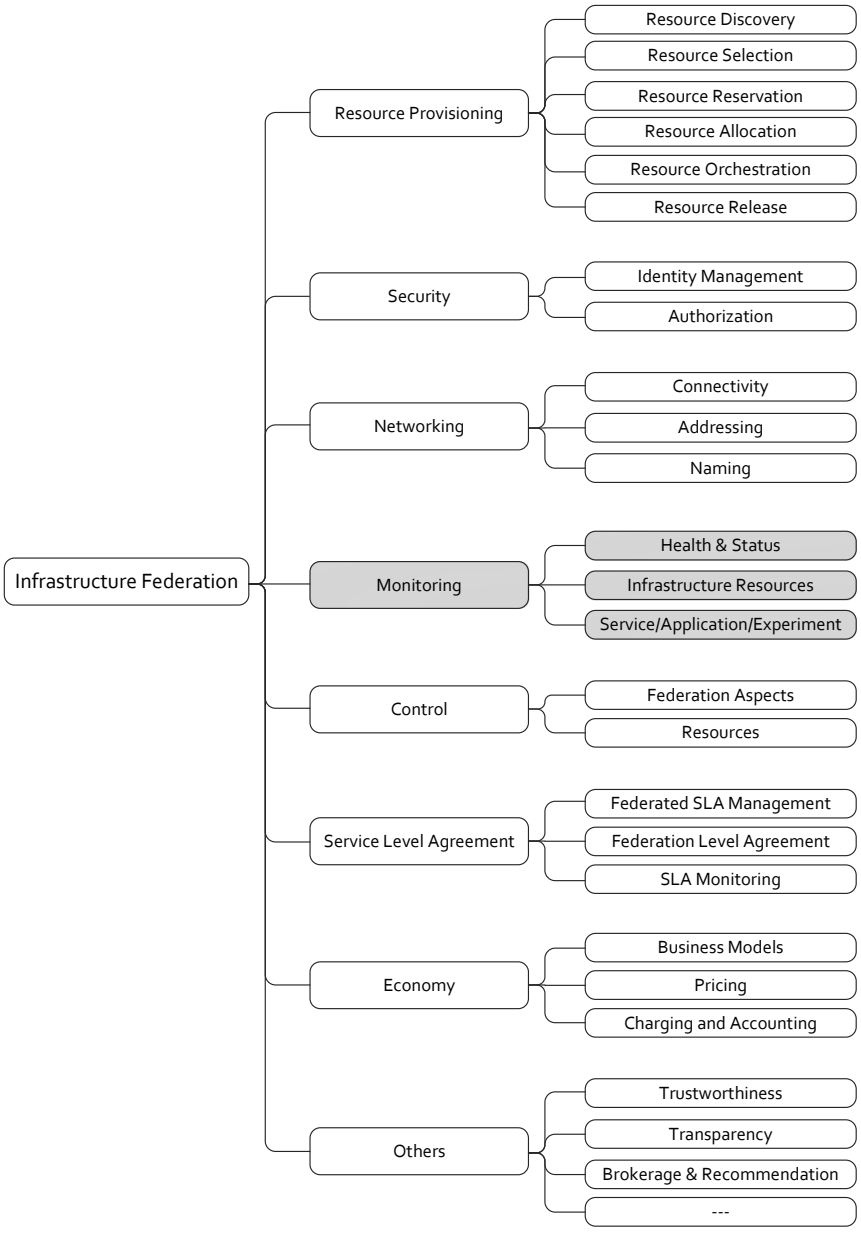


Fig. 1.4.: Major aspects of a federated infrastructure (based on [39])

Figure 1.5 represents a high-level overview of a reference architecture taken as an example for a possible infrastructure federation, where some main components, stakeholders and APIs are illustrated. Using a suitable tool, that in return uses common federation APIs, users can discover and request resources including monitoring services they are interested in. They get monitoring data through a common monitoring API to be used for various purposes, e.g. performance analysis of services and applications being used or tested, tracking the behavior and utilization of the deployed resources, and accordingly controlling them. For simplifying the figure, four

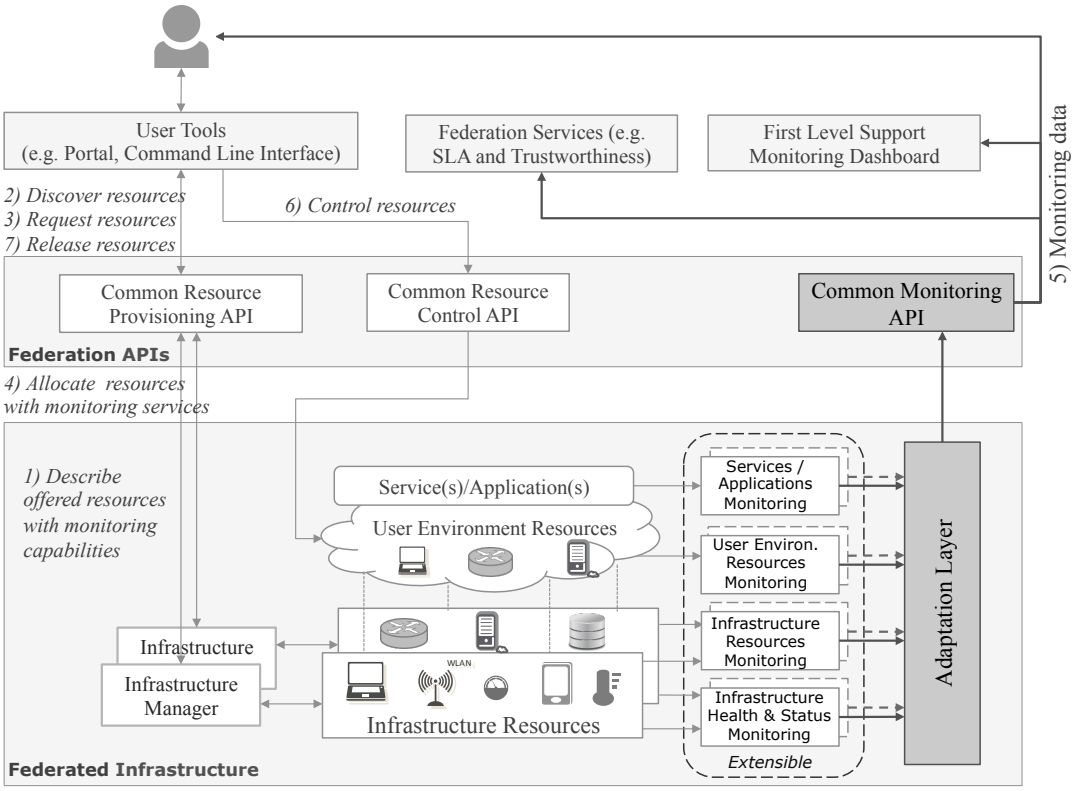


Fig. 1.5.: Scope overview of the thesis (in gray) in a reference architecture for a possible federation

main monitoring areas of applications (infrastructure health and status, infrastructure resources, resources provisioned by users to build their own environments, and users' applications and services) and their stakeholders (e.g. users, federation services, First Level Support (FLS) monitoring dashboard) are represented. The dotted boxes stand for the capability of accommodating further tools or even monitoring new areas of applications besides those already represented.

However, the scope of the thesis is depicted in Figure 1.5 in dark gray. The monitoring solutions used at the infrastructure level are out of scope. This thesis will not focus on a specific type of measurements or monitoring, or even develop new solution specific to a particular application area. Infrastructure administrators can choose the monitoring solutions that best fit their needs.

Furthermore, the focus of this thesis according to the taxonomy illustrated in Figure 1.2 comprises the classifications highlighted in gray. The solution presented in this thesis will operate across federated, *distributed* infrastructures (*multi-domain*) to provide a set of monitoring services (e.g. health and status information, infrastructure resources monitoring, SLA and trustworthiness related metrics). The data is produced and collected at the infrastructure level, converted by suitable *adapters* at the

adaptation layer, to be then provided in a *unified data representation* via a *common API* to its consumers such as federation services (deployed in either distributed or centralized modes), a central health monitoring dashboard and distributed users.

The main aims of this thesis are as follows:

- Classification and discussion of several measurement and monitoring tools and frameworks for monitoring federated infrastructures, in particular cloud platforms and test environments for FI research.
- Analysis of existing approaches for monitoring federated domains in terms of functionality.
- Design and specification of basic core functionalities for provisioning various types of monitoring information, expanding from low-level infrastructure resources up to application and services that are deployed across federated infrastructures. Such information is provided for different types of users in a unified manner through a common and standardized interface.
- Providing reference implementation of the core functionalities.

According to the problem stated and the objectives addressed in this thesis, its major contributions include the following:

- Design and develop a generic, adaptable, flexible and extensible Monitoring Architecture for Federated heterogeneous Infrastructures (MAFIA). It includes a method for unification of monitoring interfaces through a monitoring adaptation layer on top of monitoring systems deployed at the infrastructure level. This layer is responsible for providing the data in a unified representation to its users through a common monitoring API.
- Develop an ontology-based information model for measurement and monitoring in federated infrastructures. It will be generic enough to be used for the common, relevant monitoring concepts and relationships within the context of the application areas of this thesis, in particular cloud and virtualized infrastructures as well as infrastructures for FI experimentation.
- Provide a prototype implementation and disseminate the results through scientific publications, workshops and contribution to related standardization bodies.
- Prototype validation through selected use-case projects.
- Integration of theoretical concepts and practical implementations into the Future Seamless Communication Playground[9] (FUSECO PG).

[9] http://www.fuseco-playground.org

1.5. Methodology

This section outlines the methodology followed to achieve the objectives of this thesis.

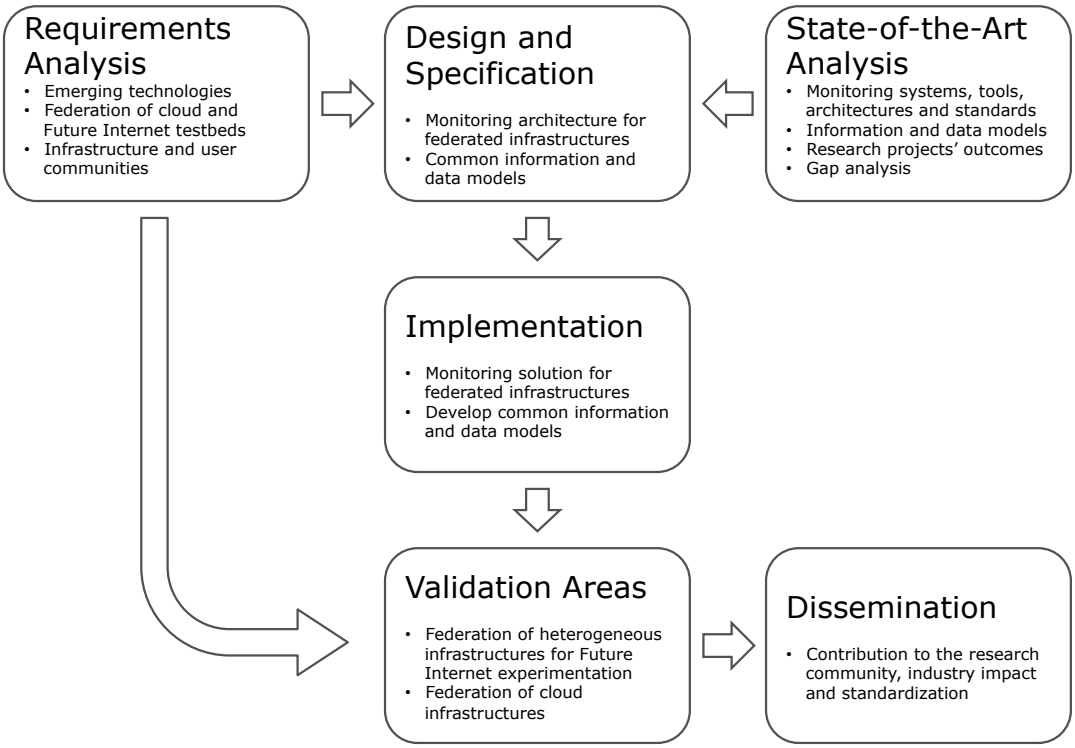


Fig. 1.6.: Methodology of the thesis

Figure 1.6 illustrates the main methodology of the work, starting with requirements analysis from main stakeholders, emerging relevant technologies (e.g. virtualization and Cloud Computing) and management systems in the fields of Cloud and FI testbeds, and accordingly performing a thorough analysis on state-of-the-art research and practice on the topics addressed throughout this thesis. The topics that influence this thesis are depicted in Figure 1.7. Then, design and implement action of the core functional elements of the targeted architecture for monitoring federated environments is presented. The major emphasis is on unification of heterogeneous interfaces of monitoring systems through developing adaptation and conversion mechanisms, as well as common information and data models. The implemented solution is validated and evaluated in two main application areas: Cloud Computing and FI testbeds. Of course, the requirements are validated. Finally, the research results are disseminated.

The methodology used in this work follows an agile model, in which the work is performed in iterations depending on multiple development cycles planned within each application area.

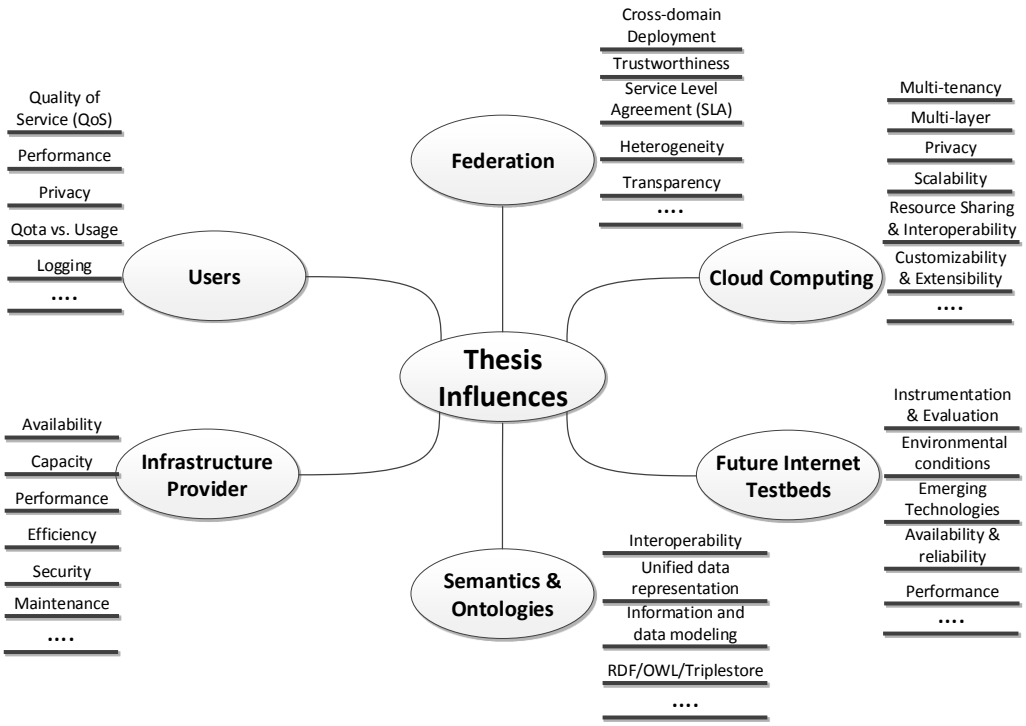


Fig. 1.7.: Thesis influences

1.6. Thesis Outline and Structure

The rest of the thesis is outlined and structured in a further seven chapters as follows.

Chapter 2 – State of the Art discusses and analyzes the state-of-the-art in monitoring methods, tools, frameworks, and information and data models used in the relevant domains, like distributed computing systems, High Performance Computing (HPC), Grid and Cloud Computing, and FI testbeds. Furthermore, outcomes of relevant research projects are considered.

Chapter 3 – Requirements Analysis includes general and specific requirements from different actors concerned in the research areas covered by this thesis. This includes high-level technical, functional and non-functional requirements from different stakeholders and users in the Cloud and FI testbeds and a federation thereof. A discussion is included of the specific requirements of a federation, namely those supporting key control and management functionalities. A gap analysis between the requirements and the state-of-the-art discussed in Chapter 2 is covered, as well as a summary of the common requirements.

Chapter 4 – Architecture Design and Specification describes in detail the design and specification of the core functionalities of the monitoring architecture

MAFIA in order to fulfill the requirements outlined in Chapter 3. The design and specification includes supporting different kinds of monitoring services, taking into account several considerations, e.g. compatibility and interoperability, standardization efforts, and minimizing the integration efforts.

Chapter 5 – Information Model describes the design and specification of the ontology-based information model, its main concepts, the individual monitoring models and the data model that are utilized by MAFIA allowing exchange of data in a unified data representation.

Chapter 6 – Implementation presents the implementation of the individual functional elements of the designed and specified architecture. This chapter discusses in detail the technologies, protocols and software used in the prototype implementation. The adoption and implementation of the developed monitoring ontology is also discussed.

Chapter 7 – Validation and Evaluation discusses the validation and evaluation of the architecture designed in Chapter 4, with the associated ontology in Chapter 5, and implementation in Chapter 6. The discussion includes a comparison between the implementations and the requirements discussed in Chapter 3. The validation of the implementations carried out at several testbeds and projects is presented.

Chapter 8 Conclusion and Outlook concludes the work through a summary, impact and outlook.

2 State of the Art

THIS chapter presents the state-of-the-art concepts, technologies and solutions in the fields related to this thesis. It first discusses the Cloud Computing model and FI research and experimentation initiatives. It then explains the federation concept and its models within these domains. Finally, monitoring solutions along with their associated data modeling and limitations are reviewed.

2.1. Service-Oriented Interaction

Service orientation is a design pattern that allows the utilization of the offerings of distributed enterprise systems in a service oriented form. It is more a concept than a technology. It is based on the provider-consumer model as depicted in Figure 2.1, which shows a simple interaction model, where a service is being offered by a provider and utilized by a consumer.

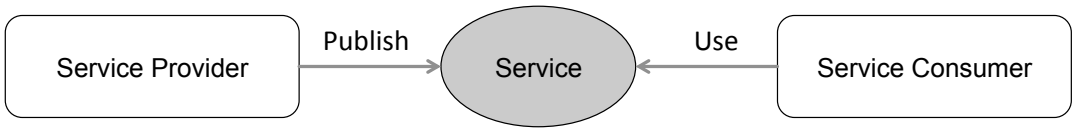


Fig. 2.1.: Concept of service orientation

Service Oriented Architectures (SOA) is an architectural style that supports the service orientation concept. In SOA, computing, network and software resources are made available for consumers as individual services or capabilities. SOA is a design paradigm for linking computational resources on demand to achieve the desired results for service consumers [40]. SOA as defined by the Organization for the Advancement of Structured Information Standards (OASIS) is: "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use services or capabilities to produce desired effects consistent with measurable preconditions and expectations [41]". In this perspective, capabilities or services are first advertised in a common way to be then discovered by consumers and finally used.

Service orientation is a relevant concept in the use case domains addressed within this thesis, namely Cloud and FI testbed infrastructures, where resources, capabilities and services are described and advertised in a common way that can then be accessed by users according to specific policies. More precisely, SOA shares concepts of service orientation with the Cloud Computing model that offers a diverse set of services in a service-oriented manner [42]–[44]. Cloud Computing is seen by [42] as a flexible platform for service providers to build their SOA solutions, while according to [43], [44], both models complement each other. Concerning the FI testbeds that

are discussed in detail in Sec. 2.3, many deal with abstraction and modularity of current and future networks and IT infrastructures in order to offer various services or capabilities for users to develop and trial novel applications and services. Such services or capabilities on top of converging networks are abstracted in a service-oriented manner allowing for cross-layer service and network composition, even across federated domains [45], [46]. In this perspective, service orientation concepts influence the design and implementation of the architecture delivered by this thesis.

2.2. Cloud Computing

Cloud Computing is seen as a new business computing model that has changed the way computing resources are consumed. It changes the focus from investments on building own infrastructures for offering functionalities to the contracting of a third-party to deliver these functionalities over the Internet on a pay-per-use basis. This model is seen as a powerful shift for computing, towards a utility model like the telephone system or even the Internet itself [47]. In this perspective, cloud computing shifts the computing from local, own infrastructures to distributed, virtual ones. Thus, computing, storage and network resources can be utilized on-demand by anybody, at any time and from anywhere in a form of Anything-as-a-Service (XaaS) following the pay-per-use utility model.

To achieve such a capability, Cloud Computing relies on dynamic resource sharing, virtualization and on-demand service provisioning mechanisms. As a matter of fact, such a paradigm enables efficient use of resources and ease of service delivery, as well as reduction of capital expenditures and operating costs.

To further clarify the picture more, infrastructures (e.g. datacenters) are usually underutilized most of the time. Thus, resource are over-provisioned, resulting in wasted investments. Furthermore, the unused resources also consume energy. These resources can be customized for peak loads (e.g. Christmas business) or planned future events (e.g. deploying services for specific period of times). That means service providers, who have small private infrastructures, can utilize/rent unused resources of other infrastructures during peak load times.

Cloud Computing has been defined differently in accordance to the context it's applied in. However the widely adopted definition given by the US National Institute of Standards and Technology [50] (NIST) is as follows:

Definition of Cloud Computing: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

The following sections describe the five cloud characteristics, as well as its service models and deployment models.

2.2.1. Cloud Characteristics

The features of Cloud Computing are diverse, including dynamic and elastic provisioning of resources and IT services in highly scalable and reliable infrastructures with QoS guarantees to meet user requirements. However the five main cloud characteristics according to NIST definition are as follows:

- **On-demand Self-service** A cloud consumer can unilaterally provision computing capabilities (resources) as needed automatically without requiring human interaction with service providers.
- **Broad Network Access** Capabilities are accessible over the network through standard mechanisms using any device (smart phones, laptops, tablets, etc.).
- **Resource Pooling** Resources offered by cloud providers are pooled to serve multiple consumers and with different physical and virtual resources dynamically provisioned according to demand. Consumers have generally no control or knowledge over the exact location of the provided resources, but might have the possibility to specify location at a higher level of abstraction (e.g. country, city, or datacenter).
- **Rapid Elasticity** The ability of the underlying infrastructure to adapt to the dynamic changes, thus allowing resources to be elastically provisioned and released. Such a capability can in some cases be supported automatically, i.e. auto-scaling up or down of resources for a given application in accordance with demand.
- **Measured Service** Automatically control and optimize resource usage, supported with suitable monitoring and reporting capabilities. Information on resource usage is provided transparently to both the provider and consumer of the utilized services.

Further economic aspects are considered. Cloud infrastructures are built, operated and maintained in a cost-effective and energy-efficient manner, benefiting from technologies like virtualization and resource sharing mechanisms [48]. Scalability and pay-as-you-go features play a role in cost reduction. Furthermore, cloud agility and deployment models support turning Capital Expenditure (CAPEX) into Operational Expenditure (OPEX), thus allowing entrepreneurs a low risk market entry. CAPEX is required to establish own, local infrastructure, but with outsourcing computational resources on an on-demand basis to other cloud services providers, those other

providers will actually outlay OPEX for provisioning capabilities, as they acquire and use resources according to operational demand [49].

2.2.2. Cloud Service Models

Cloud Computing as defined by NIST covers three fundamental cloud service models (also illustrated in Figure 2.2).

Software as a Service (SaaS) This model provides complete cloud-based, multi-tenancy applications or services using a cloud infrastructure or platform, e.g. communication services, business processes-oriented applications, games, collaboration software and tools. SaaS is defined by NIST as follows: "The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a Web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings."

Platform as a Service (PaaS) This model provides computational resources through a platform (including software, middleware, databases and development tools) to application and service developers (also referred to as cloud consumers), on which applications and services can be developed and hosted. PaaS is defined by NIST as follows: "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

Infrastructure as a Service (IaaS) This model provides managed and scalable computing, storage and network resources as services on demand in an elastic manner to the users (cloud consumers). IaaS is defined by NIST as follows: "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)."

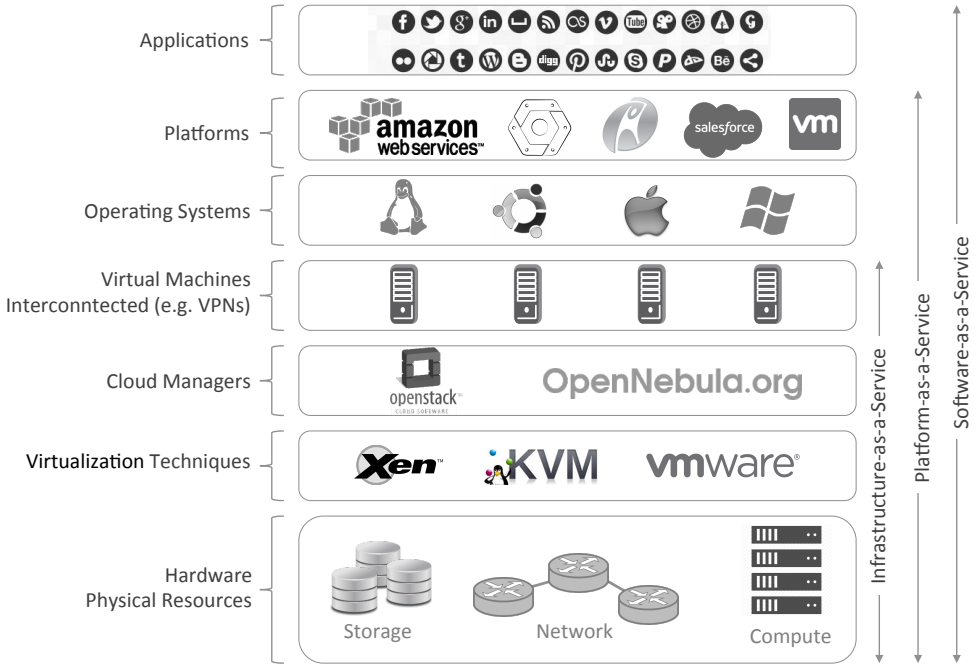


Fig. 2.2.: Cloud software stack with cloud services models

2.2.3. Cloud Deployment Models

Cloud services can be implemented in four different deployment models [50].

Private Cloud The cloud services and infrastructure are maintained on a private network, i.e. greatest level of security and control compared to other models that have potential security and intellectual property rights issues. Private Cloud can be managed and operated internally or by a third-party and hosted internally or externally. Private Cloud is defined by NIST as follows: "The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises."

Community Cloud The cloud services and infrastructure are exclusively used by a specific community of consumers or organizations that have shared concerns and interests. Community Cloud is defined by NIST as follows: "The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises."

Public Cloud The cloud services and infrastructure are provided off site over the Internet and thus made available to the general public or a large industry group provided by an organization selling cloud services. Public Cloud is defined by NIST as follows: "The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider."

Hybrid Cloud The cloud services and infrastructure are built in a composition of two or more different cloud models (private, community, or public) that remain individually unique entities but together form a hybrid cloud. Through such a model, cloud infrastructures can perform on-demand, on-peak out- and in-sourcing provisioning of IT resources either partially or completely. Hybrid Cloud is defined by NIST as follows: "The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)."

2.2.4. Virtualization and Multi-Tenancy

Virtualization recently become a significant buzzword in the industry, in particular with the introduction of Cloud Computing. However, virtualization as a concept is not new. It has been employed since the days of mainframe computers through the introduction of the concept of multitasking in Operating System (OS) [51]. Virtualization can be defined as the creation of logical (virtual) instances from physical ones. In other worlds, it is the abstraction of physical hardware to appear as multiple logical instances. Virtualization technology enables sharing (splitting) or combination (aggregation) of computing resources to offer virtual ones.

It is considered as a key enabling technology for Cloud Computing and many other services (e.g. NFV). Examples of the areas and services empowered by virtualization are illustrated in Figure 2.3. Its power can reduce both capital and operational expenses through less and efficient resource utilization and energy saving. Its power supports achieving higher levels of network and service flexibility and resilience, e.g. high availability and scalability of infrastructure resources as well as supporting load-balancing and disaster-recovery through fast and simple deployment and migration of virtual instances.

Virtualization is applied in different domains: server (also called system) virtualization [51], storage virtualization, network virtualization [52]–[55], network interface cards [55]–[57] and network service virtualization (called also NFV) [58]. However, server virtualization is the most relevant technique in Cloud Computing and NFV. It enables the creation and runing of multiple (virtual) server instances on the same

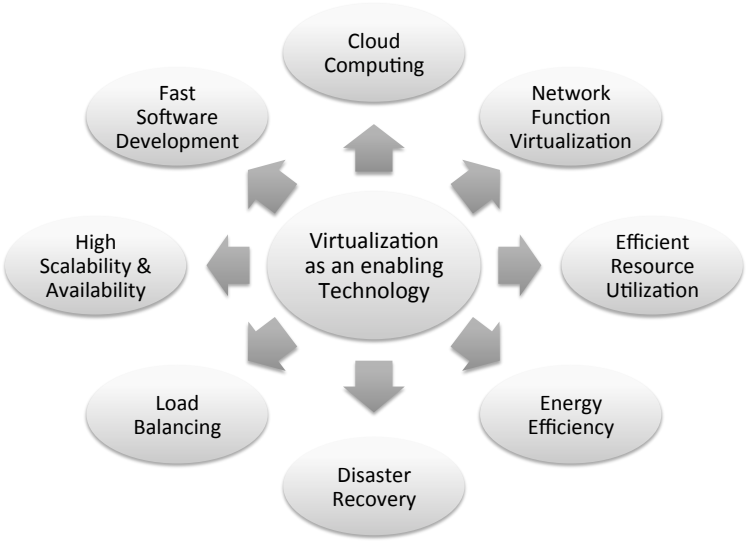


Fig. 2.3.: Benefits of virtualization technology

physical server hardware, while each server instance (called Virtual Machine (VM)) runs separately on its own OS as if it were a physical server. Physical resources (memory, CPUs, disks, etc.) are shared by the created VMs which can be rebooted independently. Server virtualization is based on virtualization software commonly known as *Hypervisor*, also called *Virtual Machine Manager (VMM)*. It allows multiple OSs to run concurrently on a host server. It is called as *hypervisor* because it is conceptually one level higher than a supervisor. It provides several management functions: creating, starting, moving, copying, pausing, stoping and destroying VMs, as well as managing the execution of guest OSs.

Different types of server virtualization exist [51]. These are illustrated in Figure 2.4 and explained as follows:

- *Full Virtualization* Provides total abstraction of the underlying hardware, so that each VM has dedicated hardware resources: memory, CPU, disk, etc. VMs are isolated and each can run any kind of OSs (e.g. Linux, Windows, Mac, etc.) that is referred to as *guest OS*. Examples of hypervisors from this type are XEN[1] and Kernel-based Virtual Machine (KVM)[2].

- *Paravirtualization* As some hardware platforms don't support virtualization, this type of virtualization requires modifications to the guest OS to run in this virtual environment. However, like full virtualization, guest OSs are executed in their own isolated VMs. There are some versions of XEN that support this type of virtualization.

[1] http://www.xenproject.org
[2] http://www.linux-kvm.org

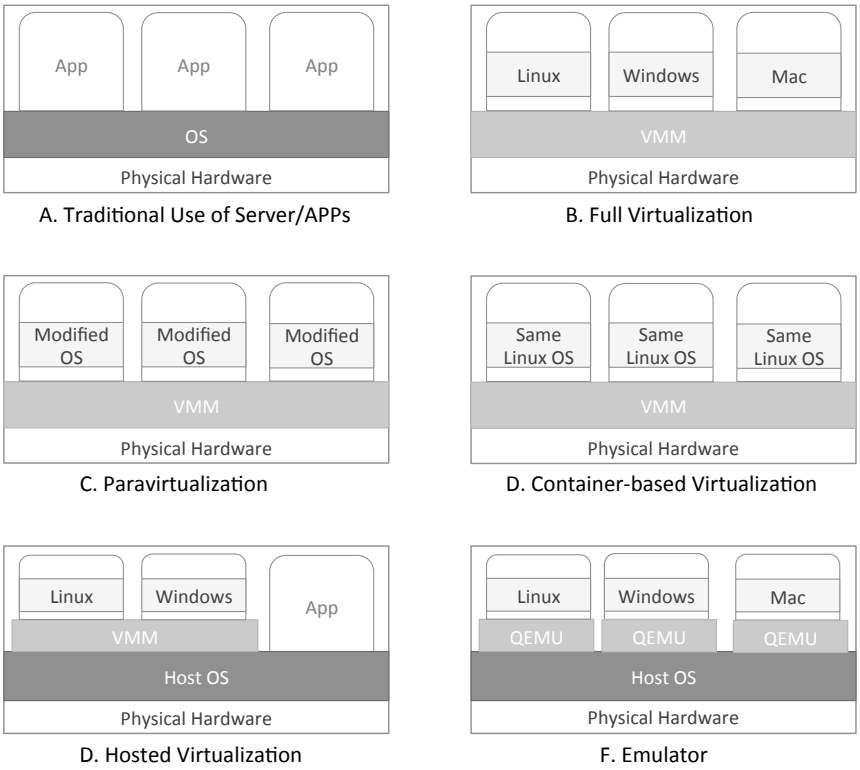


Fig. 2.4.: Types of server virtualization

- *Hosted Virtualization* The hypervisors runs as a process within an OS that is run on the physical hardware and called *host OS*. The hypervisor does not need its own drivers, but uses those provided by the host OS. VMs are isolated and each can run any kind of OSs. Examples of the VMM from this type include VMWare Workstation[3], VirtualBox[4].

- *Container-based Virtualization* This type of virtualization creates multiple isolated containers. Each container performs and executes exactly like a stand-alone server. All containers use the same Linux kernel, however, they can be rebooted independently and have root access. Examples of hypervisors of this type are OpenVZ[5] and Linux-Vserver[6].

- *Emulator* A machine emulator like QEMU[7] emulates its own hardware, and thus, enables software that was compiled for a specific architecture to work on a different architecture. It supports virtualization when executing under XEN or KVM.

[3] http://www.vmware.com
[4] https://www.virtualbox.org
[5] https://openvz.org
[6] http://linux-vserver.org
[7] http://wiki.qemu.org

Multi-tenancy is another technique that has become prominent in cloud domains although the concept is not new [59]. The concept refers to the idea of running multiple independent instances of one application simultaneously to serve multiple users. Such instances are referred to as *tenants*. A *tenant* is a logically isolated space dedicated to one or more users (organization). *Multi-tenancy* is based on sharing infrastructure resources and services among tenants that needs to be managed, maintained and made available in multiple isolated instances [59], [60].

However, authors in [61] extend the multi-tenancy definition to address the whole technological stack behind the different cloud service models (SaaS, PaaS, and IaaS), and therefore, defined multi-tenancy as "the sharing of the whole technological stack (hardware, OS, middleware and application instances) at the same time by different tenants and their corresponding users. The differentiation between tenants (organizational domains) and users (individual entities inside these groups) allows for different levels of granularity in the sharing of resources [61]."

Multi-tenancy model is meant to support per-tenant customizable applications with different SLAs that have to be met. Multi-tenancy has different level of isolation that can be realized through virtualization (in forms of VMs), Docker[8] or Linux Containers[9] (LXC). However, as virtualization is the dominant technique used in Cloud Computing, multi-tenant application deployment has advantages in terms of scalability, reducing costs and time-to-market as organizations focus only on their applications and not the infrastructure itself, although some concerns, such as security, remain the same.

2.2.5. Cloud Management Tools and Standards

Cloud Computing has been gaining more and more attention in recent years, and has become an integral part of IT infrastructures. As a matter of fact, a range of global initiatives has been established from both industry and academia, spanning from cloud management APIs and open source toolkits to specifications within several Standards Developing Organizations (SDOs).

The fundamental component of cloud management systems is the virtualization manager, which is in charge of the abstraction of the underlying physical resources, and thus, also in charge of managing the provision of virtual instances. Such a manager can work with one or multiple hypervisors (Xen, KVM, VMware, etc.). For users to interact with the manager cloud management APIs and de-facto standards have been defined in order to discover, deploy, stop and destroy VMs associated with the required storage and networking resources.

The Open Cloud Computing Interface [197] (OCCI) standardized by the Open Grid Forum (OGF) is a protocol and API for cloud management. OCCI initially focused

[8] https://www.docker.com
[9] https://linuxcontainers.org

on the IaaS model for remote management interactions with cloud infrastructures, but recent releases include other models (e.g. PaaS and SaaS). Similar to the OCCI, the Distributed Management Task Force (DMTF) worked on the Cloud Infrastructure Management Interface [175] (CIMI), which is an open standard API specification for managing cloud infrastructures. Likewise, the Open Virtualization Format [206] (OVF) is an open format for packaging software to be run in VMs. To manage storage resources, the Storage Networking Industry Association (SNIA) defined the Cloud Data Management Interface [174] (CDMI). The OCCI is compatible with both OVF and CDMI. Another standard is the Amazon Elastic Compute Cloud (Amazon EC2)[10] Web service interface that is owned and used by Amazon, but is widely adopted by many cloud service providers for managing and controlling resources.

In addition to these standards, OASIS has defined the Cloud Application Management for Platforms (CAMP) specification that aims at enhancing the interoperability of interfaces of different cloud platforms. As CAMP has limited functionalities to support portability and orchestration of applications across platforms, OASIS has defined the Topology and Orchestration Specification for Cloud Applications [222] (TOSCA). TOSCA aims at leveraging portability of application layer services, as well as the deployment of complex service topologies and their orchestration across cloud environments [62].

A range of cloud management solutions exists from both industry and academia. Examples include, OpenStack[11], OpenNebula[12], Eucalyptus[13]and VMWare[14]. Most relevant to this thesis is OpenStack, which is becoming more and more the dominant de-facto standard. Its architecture comprises several components with different functionalities. The essential ones are *Nova* (a Cloud Computing fabric controller that manages VMs), *Glance* (supporting disk and server images discovery, registration and delivery), *Neutron* (network management including OpenFlow control), *Swift* (object storage), *Cinder* (block storage), *Keystone* (identity management) and *Ceilometer* (providing metering and statistics from OpenStack components).

Finally, concerning cloud infrastructures and services monitoring, there are no standards or architectures. An exception is the extension to OCCI, which supports the management of monitoring setup [63] and is similar to the research done at the early stage for this thesis [64]. However, state-of-the-art solutions on monitoring distributed systems and grid infrastructures were initially used for this purpose. But as these are limited in their functionalities and not convenient to serve the requirements of clouds, new solutions and architectures are under development. These are discussed in Sec. 2.5.

[10] http://aws.amazon.com/ec2
[11] https://www.openstack.org
[12] http://opennebula.org
[13] https://www.eucalyptus.com
[14] https://www.vmware.com

2.3. Future Internet Experimentation

The growing success and increasing confidence in the Internet in our daily lives have led to major debate among experts on the ability of the current architecture to cope. Some believe it may collapse under increasing demands of future applications [5], [65], as it was not designed in the beginning for the current level of usage [4], [5]. David D. Clark, MIT, in an article in *MIT Technology Review* in 2005, said "The Internet is broken". It has become a complex ecosystem that is rapidly growing [1]–[3] at a great pace of change and uncertainty in many of its areas.

As a matter of fact, several global activities propose FI architectures (following either clean-slate or evolutionary Internet design approaches) to solve the limitations of the current Internet [4], [5], [66]. Others have introduced new technologies and future architectures [6] to cope with the management complexity of the Internet and networks in terms of improving QoS, resource utilization, reliability, security and reduction of CAPEX and OPEX costs.

Emerging communication and networking paradigms and technologies have been introduced to overcome some of the aforementioned issues of the current Internet, e.g. virtualization, Cloud Computing, SDN, NFV, M2M and IoT . However, they bring forth new business models and opportunities for building Smart Cities, eHealth, eGovernments, eLiving, and other services and applications.

For such new services and applications to be trialed and evaluated at scale, convenient experimental environments that support, or are enabled with, the aforementioned paradigms and technologies are required. This is because the transition from theoretical or simulation based research into production is not always the optimal strategy, especially if the newly developed services or protocols will be deployed in large-scale or across heterogeneous networks. It is therefore foreseen that experimentally driven research conducted on large-scale and real-world facilities is essential for FI research and development. This opens the door for researchers, applications developers, and SMEs to study and test their new ideas and products in real-world, controllable, and cost-effective environments. Additionally, allowing various stakeholders (e.g. technology and application developers, platform providers, integrators, and end-users) to meet in such environments where they can influence each other's work.

Many research activities worldwide are focusing on the definition and the development of FI architectures. Some of them even build suitable experimental facilities. The major examples are GENI and the Future Internet Design[15] (FIND) programs in the US , the European FIRE initiative and FI-PPP program, and the G-Lab in Germany. Similar programs have been launched in Asia as well. Examples include AKARI[16] in Japan; the Joint Asian activities that are carried out under the

[15] http://nets-find.net
[16] http://akari-project.nict.go.jp

Asia-Pacific Advanced Network (APAN)[17] initiative; the Asia Future Internet Forum (AsiaFI)[18]; the Japan Gigabit Network (JGN)[19] testbed and StarBED (Hokuriku Research Center)[20] emulation testbed that support implementation and testing of Next-Generation Network (NGN) technologies; and the KREONET (Korea Research Environment Open NETwork)[21], which is a national R&D network managed by KISTI (Korea Institute of Science and Technology Information) connecting twelve regional network centers and over 200 participating institutions.

2.3.1. Global Environment for Network Innovations (GENI)

GENI is a collaborative program funded by the US National Science Foundation (NSF). It started in 2007 with the aim of providing a global large-scale experimental platform for testing and validation of FI architecture. It relies on the concept of combining heterogeneous, virtualized resources to produce a single platform for multiple network researchers simultaneously. Its current status is beyond its initial prototyping stage, supporting wider deployment of increasingly standard technologies such as SDN, OpenFlow and WiMAX [13].

Its high-level architecture comprises multiple building blocks and functionalities, the major ones being components, aggregates, slices and a clearinghouse [67]. The components refer to the offered resources, while a group of components that are owned and administrated by an organization are referred to as an aggregate. A slice represents a set of slivers (virtual instances or part of specific resource) spanning a set of network components. Offerings are available for experimenters through a control framework run by a clearinghouse, which is a collection of related services supporting federation among experimenters and aggregates. Within the context of GENI, multiple testbeds are federated to build the GENI infrastructure. They are controlled by different competing control frameworks: the DETER Federation Architecture [177] (DFA), PlanetLab, ProtoGENI [210] (ProtoGENI), Open Resource Control Architecture [205] (ORCA) and Open Access Research Testbed for Next-Generation Wireless Networks [204] (ORBIT). More details about these control frameworks are given in [13], [67].

GENI adopts a federation model that includes resources owned and operated by different platform providers [13]. It aims at providing a coherent, large-scale facility that embraces contributions from a range of autonomic networks. GENI federates at the international level with other testbeds, e.g. with the KREONET testbed in Korea [68].

[17] http://www.apan.net/
[18] http://www.asiafi.net/
[19] http://www.shiratori.riec.tohoku.ac.jp/jgn.html
[20] http://starbed.nict.go.jp/en/
[21] http://www.kreonet.net

2.3.2. Future Internet Research and Experimentation (FIRE)

FIRE is a European initiative that was established by the European Union's Seventh Framework Programme for research and technological development (FP7) in 2007. It aims at creating an open environment for research and development within the context of FI to study and trial new concepts and ideas through conducting large-scale experiments on new paradigms, technologies and networking concepts and architectures [65].

Under the umbrella of FIRE, several research projects[22] has been conducted, as well as multiple testbeds established in various domains. In addition to these, FIRE encompasses support activities that deal with socio-economic and sustainability aspects.

As the initiative started some years ago, several projects have already closed that focused on researching FI concepts, e.g. OneLab[23] [200] (OneLab). OpenLab[24] [203] (OpenLab) provided testbed federation model proven through the federation of PlanetLab Europe with the global PlanetLab infrastructure that is practically realized through the OneLab2 [200] (OneLab2). ResumeNet[25] studied network resilience aspects in FI and future networks. There are also several others FIRE projects[22].

Considering the outcomes of the closed FIRE projects, new projects have been started that have built experimental facilities for FI experimentation. Examples include Building Service Testbeds on FIRE [17] (BonFIRE), OpenLab, SmartSantander[26] (SmartSantander), OpenFlow in Europe: Linking Infrastructure and Applications[27] (OFELIA), Cognitive Radio Experimentation World[28] (CREW), Community Networks Testbed for the Future Internet[29] (CONFINE), Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures[30] (FEDERICA), EXPERIMEDIA[31] (EXPERIMEDIA), Federation for FIRE[32] [11] (Fed4FIRE) and more[22].

In addition to all these projects and testbeds, several Coordination and Support Actions (CSAs) have been established to coordinate these efforts as well as to enhance the exchange of information between them. Succeeding the closed activities PAR-

[22] http://www.ict-fire.eu/home/fire-projects.html
[23] http://onelab.eu
[24] http://www.ict-openlab.eu/project-info.html
[25] http://www.resumenet.eu
[26] http://smartsantander.eu
[27] http://www.fp7-ofelia.eu
[28] http://www.crew-project.eu
[29] https://confine-project.eu
[30] http://www.fp7-federica.eu
[31] http://www.experimedia.eu
[32] http://www.fed4fire.eu

ADISO[33], FIREworks[34], FIREBALL[35], and MyFIRE[36], the new activities FUSION[37], FIRE STATION[38], AmpliFIRE[39] and CI-FIRE[40] have been established. The last two support the FIRE community to prepare FIRE for 2020. Both aim at developing a sustainable vision and business models, as well as the role of FIRE facilities beyond the funding phases.

As BonFIRE, OpenLab and Fed4FIRE are the most relevant projects to this thesis, they are discussed further below.

BonFIRE The BonFIRE project has designed and built a multi-site cloud testbed, targeted at the Internet of Services (IoS) community. It supports large-scale testing of applications, services and systems over multiple, geographically distributed, heterogeneous cloud testbeds. Its aim is to provide an infrastructure that allows experimenters to control and monitor the execution of their experiments to a degree that is not provided or might not be found in traditional cloud infrastructures. Among its key functionalities are controllable resource management and configuration (including the ability to emulate network parameters), in-depth monitoring of virtual and physical infrastructure metrics, elasticity support, user-friendly experiment descriptors and scheduling [17].

OpenLab The aim of OpenLab was the design and development of a federation framework operating across heterogeneous domains (wireless and wired), allowing researchers and engineers to execute experiments at larger scale. It provides advances on early prototypes serving the demands of FI services and applications. The OpenLab federation framework facilitates the federation at the the level of control, experimental, and data planes. The control plane is in charge of the management of the underlying testbeds resources, which includes resource virtualization, inter-resource communication, etc. The experimental plane includes all software components that allow users to execute experiments with a range of monitoring and measurement capabilities. The data plane is concerned with measurements and data collection, i.e. ensuring the interoperability of tools and data repositories [69].

Fed4FIRE The aim Fed4FIRE is to build the largest federation of FI experimentation facilities in Europe covering different technologies, domains, services and user communities. A large number of testbeds are available, and these have diverse

[33] http://paradiso-fp7.eu
[34] http://www.ict-fireworks.eu
[35] http://www.fireball4smartcities.eu
[36] http://www.my-fire.eu
[37] http://www.sme4fire.eu
[38] http://www.ict-fire.eu/home/firestation.html
[39] http://www.ict-fire.eu/home/amplifire.html
[40] http://www.ci-fire.eu

focuses, such as network management and networking protocols (e.g. OFELIA), wireless cognitive networks (e.g. CREW), Cloud Computing and IoS (e.g. BonFIRE), IoT (e.g. SmartSantander) and content-centric networking and peer-to-peer (e.g. EXPERIMEDIA). The idea is to build an even larger and more powerful facility beyond initial federations within the context of FIRE (e.g. OneLab, OneLab2 and OpenLab) that had smaller or different levels of integration and complexity. Nevertheless, the establishment of such a federation is not a trivial task, as it takes time and effort to define the right approach and to build an ecosystem in which experimentation facilities can coexist without blocking innovation. Fed4FIRE builds further on the technologies and approaches developed in OpenLab. It draws on federation experience built in other projects like OFELIA, BonFIRE, and CREW. It generalizes these tools for the FIRE community as a whole. Fed4FIRE follows a heterogeneous federation where all testbeds run their native testbed management software but adopt a set of APIs on top of the federated infrastructure [10].

2.3.3. Future Internet Public Private Partnership (FI-PPP)

FI-PPP is a European programme for FI innovation. It is a more market-oriented program aimed at accelerating the development and adoption of FI technologies in Europe, advancing the European market for smart infrastructures. Several research projects[41] have been conducted under the umbrella of the FI-PPP program, but at its heart is the FIWARE project. This project, along with other projects that are relevant to this thesis, are discussed below.

FIWARE FIWARE delivers a service infrastructure that offers a large number of general-purpose, vendor-independent platform functions and services, called Generic Enablers (GEs). They can be seen as enabling technologies in different fields to build a sustainable foundation for the FI and Smart Cities Innovation. These are offered through the FIWARE Catalogue[42], which includes over 80 Generic Enabler implementations (GEis) that can be applied in the fields of IoT, healthcare, transports, energy and environments, media and content, manufacturing and logistics, and social and learning services. These GEs are offered by different technology providers in form of open and vendor-independent APIs. The catalogue can be enriched with additional, new GEs. Technology providers can build their technologies in any field and provide them in the form of GEs through the FIWARE Catalogue. These can be used by different application developers who can provide significant feedback to the technology providers. They can accordingly enhance their solutions.

[41] http://www.fi-ppp.eu/projects/
[42] http://catalogue.fi-ware.org

XIFI Experimental Infrastructures for the Future Internet[43] [70] (XIFI) focuses on building a sustainable pan-European federation of cloud-based test infrastructures for open innovation developments offering a large number of general-purpose, vendor-independent platform functions and services (GEs), in different fields within the context of FI and Smart Cities Innovation. XIFI federation architecture is introduced in [70], most of its components are based on FIWARE GEis and the remaining components are developed within the project. An instance of the XIFI federation is implemented and deployed though a federation of more than 17 infrastructures under the so-called FIWARE Lab[44] [19] (FIWARE Lab). Although its architecture is mainly built through using some GEis, instances of these GEis besides the remaining FIWARE GEis are all offered through FIWARE Lab for users (FI application developers).

FIWARE Lab FIWARE Lab is seen as the most suitable environment for open innovation where various communities can meet and influence each other, including infrastructure providers, technology providers and application developers. Detailed offerings and benefits for each of these stakeholders are discussed in [70]. Besides the traditional cloud resources (e.g. VMs), the strength and the unique offer of FIWARE Lab is the innovative general-purpose platform functions and services (the FIWARE GEs), extended with advanced FI facilities like sensor-enabled environments, SDN-enabled infrastructures, seamless access anywhere, anytime and from any device (Wi-Fi, 2G, 3G, 4G, etc.), and more.

Future Internet - Social and Technological Alignment for Healthcare[45] [185] (FI-STAR) Its goal is to establish early trials in the healthcare domain building on the FI-PPP technologies, i.e. on the FIWARE GEs, in particular those from the cloud and IoT domains. It delivers a suitable platform in the health sector that is built out of GEs and further FI-STAR components (referred to as Specific Enablers (SEs)), which are available through the FI-STAR Catalogue[46].

INfrastructures for the Future INternet CommunITY[47] (INFINITY) The role of the INFINITY project within the context of FI-PPP is similar to that of AmpliFIRE[39] and Ci-FIRE[40] within the FIRE program. INFINITY acts as a CSA to promote and ensure the success of FI-PPP program. It aims at capturing and communicating information about available testbeds and ICT infrastructures and making them available for diverse communities, as well as facilitating the exchange of any interoperability requirements and issues.

[43] http://fi-xifi.eu
[44] http://lab.fi-ware.org
[45] https://www.fi-star.eu/fi-star.html
[46] http://catalogue.fi-star.eu
[47] http://fi-infinity.eu

2.3.4. German-Lab (G-Lab)

The G-Lab initiative is the largest German project driving FI research in Germany and is funded by the German Bundesministerium für Bildung und Forschung (BMBF). It includes several research projects supported with a Germany-wide experimental facility investigating the interplay between new technologies and the requirements of emerging applications. G-Lab's vision is that research studies are accompanied by experimentation performed at a G-Lab facility. Its main concept is based on the tight coupling of research and real experimentation with the vision that the G-Lab platform might evolve into the FI backbone itself through adopting emerging technologies and future paradigm. From a technical viewpoint, the G-Lab facility leverages the PlanetLab central software. Concerning federation aspects, some were tackled by the G-Lab Deep project[48] from a conceptual perspective. The project also focused on service composition and security.

2.4. Federation Models and Approaches

The term 'federation' has been defined in different ways. But the common understanding of a federation is an agreement between independent entities or domains on a certain level of commitment that is ensured and managed internally. The federation in its application varies from one context to another. For instance, today's Internet services, such as cloud services and multimedia services (e.g. Internet Protocol Television (IPTV) and Voice over Internet Protocol (VoIP) that require strong guarantees on end-to-end QoS), are evolving towards complex composed services spanning across administrative domains. These services benefit diverse capabilities provided by multiple domain providers and complement each other [71]. To achieve service composition across independent domains, there is a need for coordination and collaboration agreements across them.

In general, federation agreements in the ICT world are already in use but on a limited context. Federation agreements, according to Celesti et al. [72], can be established vertically or horizontally. In the vertical federation, services or capabilities of one provider are leveraged by another. For instance, cloud providers leverage cloud resources from other cloud infrastructure providers. While in the horizontal federation, cloud providers federate among themselves to gain economies of scale, as well as enlarge or complement their offerings. The horizontal federation is the one in line with the vision and work within this thesis. Such a type of a federation is discussed in different research projects and application contexts, with examples including Cloud Computing domains [17], [47], [70], [72], [73], FI testbeds [10], [12], [13], [45], [71]–[74], and e-infrastructures [73].

[48] http://www.g-lab-deep.de/index.html

It is to be noted that neither the motivation nor the demand for federation are discussed in this thesis as these are out of scope. This thesis is built on the fact that federations exist and are in demand from various sectors (e.g. economy, e-commerce, Smart-Cities and FI services and applications) associated with a range of research projects driven by academic research and industrial needs.

The new demands and trends for federation have contributed to refine its original, simple definition as an agreement among parties to be more specific to the areas of application. Although there are multiple definitions for federation in the literature in accordance with this thesis, this thesis follows the one introduced in Sec. 1.1. To elaborate more on that definition, a federation is a model for the establishment of a larger-scale, higher performance, and diverse infrastructure through the interconnection of two or more independently administered infrastructures. This allows infrastructures owners to share resources and services for increased multilateral benefits to their users. This is achieved following a set of agreements on a certain level of commitment and operational procedures, and defining internal mechanisms to enforce these at the infrastructure level.

Multiple federation models and approaches exist. These are discussed in the following sections.

2.4.1. Federation Models

Multiple federations exist that differ from each other in their approaches, models, and the tools and interfaces used. For instance, in Cloud Computing, Toosi et al. [39] discuss several models from different perspectives. However, one of the common goals of federation activities in the context of e-infrastructure federation is the establishment of the self-sustainable operation of the federation, as operation is a critical aspect of any federation.

Within the context of FIRE, models and strategies have been analyzed, for example, within the Pan European Laboratory Infrastructure Implementation [207] (Panlab) project several relevant deliverables [75], [76] have been produced and a FIRE Office [182] was planned to be established. Such activities continued in multiple projects, but within the Fed4FIRE project, federation models and scenarios are discussed extensively from the following different perspectives:

- *federation architecture and management software* [10], and
- *federation operation and stakeholders' interaction* [77] that itself is mainly based on the federation framework defined by the Federated IT Service Management[49] [73] (FedSM) project. Based on this framework, similar analysis has been done within the context of the FI-PPP program to determine the most suitable model for its federation [70].

[49] http://fedsm.eu

2.4.1.1. Models for Federation Operation and Stakeholders' Interaction

In this section the main aspects of the FedSM federation framework are described [70], [73], [77]. The three main actors in a federation are as follows:

- *User* The consumer or customer of the federation (can also be a group) that benefits from the offered services or resources.
- *Federation members* A group of organizations or bodies that form a federation. These are the service and resource providers. In the case of the domains addressed in this thesis, they are the infrastructure providers.
- *Federator* An individual body or component that provides added-value services for the whole federation. It can either act as an advisor for the users or even has full control and management over the whole federation including the users and federation members.

These actors can interact in one of three different ways in order to utilize a service:

- *Certification authority* Users and federation members interact directly with each other. The role of the federator is to provide the means for the interaction to be realized, e.g. through defining protocols and standards.
- *Loose integration* Part of the interaction involves the user interacting with federation services provided by the federator before interacting directly with the federation member.
- *Tight integration* All interaction is between the user and the federator.

In accordance with these three different models of interactions, FedSM introduced five different federation models that vary from each other depending on the way the main actors interact among each other. These are illustrated in Figure 2.5 and described as follows [70].

- **Invisible Coordinator** The federation acts as a certification or validation authority. The federator defines membership rules and validates compliance of the federation members who work to comply with the rules and seek certifications from the federator. Users find infrastructures through other channels (e.g. search engines, marketplaces) and then request services (Figure 2.5a).
- **Advisor** The federator advises federation members on how to offer and promote their services through the federation. Users describe the required services to the federator, which advises them on where to find the needed services in the form of recommendations. Users then decide which federation members to interact with (see Figure 2.5b).

- **Matchmaker** The federator advises federation members on how to promote their services, along with terms and conditions, through the federation. Users describe the required services to the federator, which matches users' requests to the services offered by infrastructures, and thus, performs resource allocation (reservations) on behalf of the users (Figure 2.5c).

- **One-Stop-Shop** The federation provides a channel for the federation members to advertise their services. Users describe the required services to the federator, which performs resource allocation (reservations) on behalf of the users, monitors usage (based on monitoring information collected from infrastructures) and provides billing. Users pay the federator who is the initial point of contact. Federation members are still able to bill one another (Figure 2.5d).

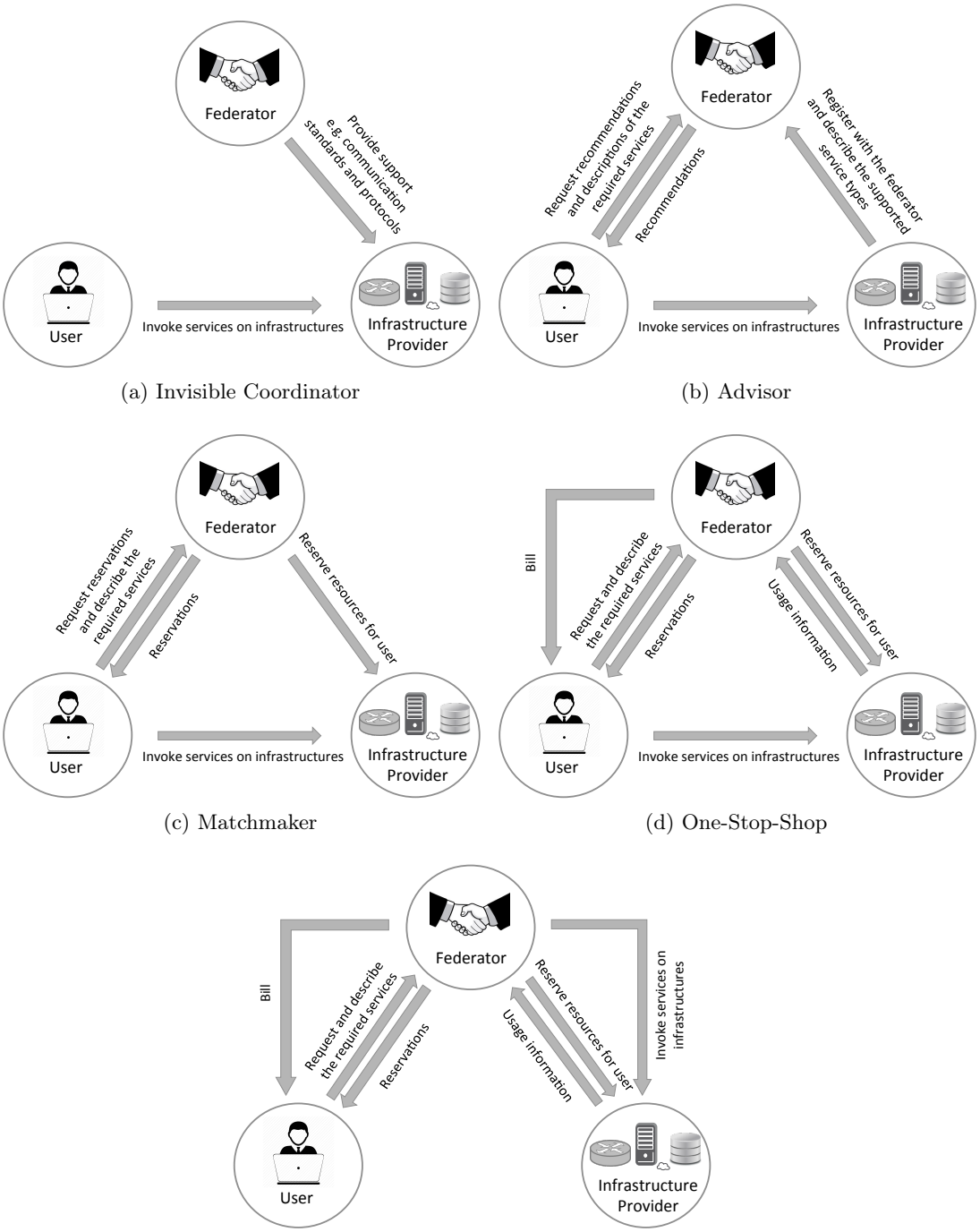
- **Integrator** The federation is in charge of all interactions with users. Users interact with the federation searching for offerings and describe the required services and resources. These are reserved and then invoked by the federator at the proper time. Billing and payment are done by the federator (Figure 2.5e).

An *advisor* or *matchmaker* model implies a *looser integration* of resources and stakeholders by the federator, which provides support to enable resource identification by users and resource exploitation at the resource provider site. An *integrator* or *one-stop-shop* by contrast increases the control and support offered by the federator in running and managing resource utilization. Their role reflects the *tighter integration* interaction model while the *invisible coordinator* corresponds to the *certification-based* interaction. Making a decision on the most suitable model for a target federation is not a trivial task. This decision would also be influenced by economic factors affecting the target federation.

2.4.1.2. Models for Federation Architecture and Management Software

Suitable mechanisms are required to allow infrastructures to interoperate in a standardized manner. Resources and services offered through the federation have to be managed at the federation level so as to appear to users as if they were provided by a single infrastructure provider. Therefore, several management mechanisms are to be specified and standardized in order to support the entire service or experiment lifecycle (resource description, discovery, provisioning, monitoring, control and release), as well as authentication and authorization aspects. From this perspective, the federation architecture has to be designed to cover the necessary mechanisms. According to a study conducted within the FIRE initiative for its testbed federation, four possible models for the federation architecture are introduced [10]:

- **A Central Management System** Resources of all infrastructures are managed by a single management system at the federation level. Infrastructures



(a) Invisible Coordinator

(b) Advisor

(c) Matchmaker

(d) One-Stop-Shop

(e) Integrator

Fig. 2.5.: Federation models according to FedSM, based on [70], [73], [77], [78]

themselves do not run local management systems. The advantage of such a system is easy deployment, management, and maintenance. On the other hand, infrastructure providers will all be forced to replace their management systems with the common one. Considerable efforts would be necessary to adopt the new software, which may not be beneficial for some platforms in terms of compatibility in case they are part of other federations.

- **A Central Frontend Point** All infrastructures and their tools are listed through a portal. Each infrastructure keeps using its management software and mechanisms. This approach is the easiest and cheapest federation approach since no efforts are required from infrastructure providers. In this approach there is no real federation, since users still need to have multiple accounts, and use different tools.

- **Homogenous Federation Running the same Management System on all Infrastructures** The same management system is to be deployed locally at each infrastructure. Each infrastructure will manage its resources independent from a central point. While it is advantageous in comparison to fully centralized systems each infrastructure is required to replace its own management system, and its users must adopt the new tools.

- **Heterogeneous Federation with Common APIs on top of Different Independent Management Systems** Each infrastructure keeps its management tools and software. Resources are managed as they are being provided and managed at the federation level through specified and standardized interfaces. This imposes the need to define and develop multiple interfaces for the individual management mechanisms, such as provisioning, control, etc. This approach makes it easy and flexible for further infrastructures to be involved in the federation. Its advantage is obvious for federated infrastructures that provide heterogeneous resources managed by different tools. Federation interfaces can be implemented with basic functionalities and then grow over time based on the support needed. On the other hand, specifying these interfaces and interoperability across infrastructures is not trivial and could be costly.

Heterogeneous federation might be the most suitable model for large and more dynamic federations with heterogeneous infrastructures. But for a small and fairly stable federation, the centralized approach works well.

2.4.2. Federation Approaches

Many approaches for ICT infrastructure federation exist in various sectors. To give an example, ICT infrastructure federation has been recognized as a promising mechanism in the cross-border trade sector to support global supply chains [79]. This

is because marketplaces are moving from traditional monolithic business models, where most of the activities were carried out locally, into multi-layer service orchestration coordinating out- and insourced services across domains. However, within the context of this thesis, the federation of Cloud and FI testbed infrastructures is of major concern. Therefore, this section discusses only the related approaches.

2.4.2.1. Cloud Federation

The steadily gaining momentum of Cloud Computing encourages cloud providers to develop or adopt new technologies that both enhance their services with a high level of customer satisfactions, and promote strategies to build new business models [64], [80]. Emerging markets spur cloud providers, which have different kinds of resources with different service quality levels, not only to compete but also to collaborate (federate) with each other. Federation of various cloud providers has several advantages for both cloud service providers and cloud service customers. Among the advantages for cloud providers are the complementation of the offered resources to improve resource utilization and the combination of multiple services in order to offer efficient end-to-end solutions. On the other hand, customers have the ability to combine resources and services from different cloud providers to create their own environments (e.g. IaaS or PaaS). Combining resources from multiple cloud providers becomes increasingly feasible for customers in order to benefit from factors like price differences, locality of resources, etc. Cloud federation provides the necessary means to compare and select the best cloud platform or service available at any given point in time. Cloud federation has a tremendous potential for the industry as an effective mechanism to increase the capacity of resources, diversity of offerings, as well as efficient and flexible use of resources – through shifting the operation of the offerings from in-house datacenter hosting towards the outsourcing of infrastructure or services to external use – while keeping costs relatively low.

However, the high heterogeneity of cloud infrastructures in terms of resources, management systems, tools, information exchange and other aspects is a problem. Therefore the concepts of standardization and common tools and APIs are becoming of high importance in order to cater for the necessary degree of interoperability and portability across the federation. Currently, there are already standardization activities and implemented cloud federation architectures that are up and running. The three-phase architecture model for cloud federation introduced in [16], [21] focuses on authentication aspects across cloud platforms. The RESERVIOIR [81] and Contrail [82] projects focus on the management of IaaS across domains. The Intercloud Architecture Framework (ICAF) [83] addresses multi-layer and multi-domain aspects referring to the NIST Cloud Computing Reference Architecture (CCRA) and International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Joint Coordination Activity on Cloud Computing (JCA-Cloud) standardization ac-

tivities. Many others are introduced in [18], [70], [80], [84]. The major approaches and those related to this thesis are discussed below.

The EGI Federated Cloud Task Force[50] focuses on standardizing some protocols and APIs for managing cloud resources across federated clouds such as the OCCI, CDMI and X.509 [85]. This activity focuses more on the user level but has not covered federation aspects at the networking level. In contrast, the Institute of Electrical and Electronics Engineers (IEEE) InterCloud approach [18] proposes a novel architecture for federating different clouds with more focus on network aspects, no matter which technologies, protocols or APIs are in use. This activity provides a detailed standard on interoperability and federation, called Standard for Intercloud Interoperability and Federation [47] (SIIF).

The cloud federation implemented within the FI-PPP program by the XIFI project and deployed through the FIWARE Lab has another approach that is based on FIWARE technologies and specifications. Through its mature architecture, the FIWARE Lab federation addresses the federation at the networking and user level as well considering different federation aspects, such as monitoring, SLA management, authentication, authorization, security, and procedures for joining and leaving the federation. Among the federation models discussed in Sec. 2.4.1, the choice made for the FIWARE Lab took into account sustainability strategies, as well as requirements of diverse types of stakeholders as discussed in [70]. The model adopted in FIWARE is a hybrid model covering aspects of the one-stop-shop and the integrator models. From the one-stop-shop model, a common advertising channel for infrastructure offerings is provided where users choose which services they want to use. From the integrator mode, the federator decides which services and resources will be used.

Another cloud federation approach is used by the BonFIRE project, where multiple, geographically distributed, heterogeneous cloud and network testbeds are federated. Its federation model is based on a central resource manager, used to expose a homogeneous API (namely the OCCI) and interact with the testbeds [80]. For those testbeds that can't speak OCCI, BonFIRE architecture includes an enactor component that adopts and translates the OCCI requests into a suitable testbed format. In addition to this standard API, BonFIRE uses the open standard OVF for describing experiments. BonFIRE multi-cloud facility is being further federated within the Fed4FIRE federation, aiming to the establish of a larger, heterogeneous FI experimentation facility. This will be discussed further in the following section.

2.4.2.2. Federation of Future Internet Testbeds

Resource and infrastructure federation has attracted much attention in the ICT world, as it's believed to accelerate the development of the FI and future networks.

[50] https://www.egi.eu/infrastructure/cloud/

Federation of multiple infrastructures enables on-demand composition and provisioning of complex, distributed services and applications across heterogeneous domains. The federation of ICT resources and infrastructures is therefore implemented by a couple of large-scale research projects through the federation of heterogeneous, powerful ICT testbeds, the major examples being the Fed4FIRE federation in Europe and the GENI federation in the US as discussed in Sec. 2.3. They aim to connect heterogeneous resources across several independent heterogeneous domains to provide larger-scale and higher performance experimentation facilities. For simplification, this thesis takes Fed4FIRE as a reference federation, as such the focus in the following will be on Fed4FIRE. Fed4FIRE is working on federating a number of European FI testbeds that are built and equipped with emerging and future technologies and communication paradigms (such as Cloud Computing, NFV, SDN, IoT, IoS, and mobile and wireless networks). These testbeds are available for researchers and developers to study and trial novel applications, services or ideas in real-world, controllable and cost-effective environments.

The federation of multiple, distributed, independently administrated testbeds is not a trivial task, as testbeds could use different approaches and tools to manage their infrastructure resources and services. Testbeds usually support multiple stages of the experiment lifecycle in a trustworthy manner, starting from resource provisioning (description, discovery, reservation and allocation), monitoring, control, and up to release. The commonly used frameworks within FIRE and GENI testbeds are Slice-based Federation Architecture [217] (SFA), cOntrol and Management Framework [198] (OMF), Network Experimentation Programming Interface [192] (NEPI), Teagle [8], [221] (Teagle), FITeagle[51] [183] (FITeagle) and ORBIT Measurement Library [199] (OML). SFA is used for resource provisioning and release, while OMF along with the Federated Resource Control Protocol [186] (FRCP), NEPI and Teagle can in addition be used for resource control. OML and others reviewed in [86] are used for experiment and resource monitoring.

In a federated environment, experiment lifecycle has to be supported in common ways federation-wide, i.e. interoperable mechanisms are required to manage experiments in such multi-domain, heterogeneous environments. The Fed4FIRE federation architecture [10], [11] follows the heterogenous federation model and defines a set of common APIs that are adopted by the participating testbeds to be federation compliant. This means that each testbed provider committed to the federation should implement the common, standardized APIs for external interactions. This approach allows testbeds to maintain tools and techniques already in place. These tools are used to manage resources and services offered to experimenters to support the whole experiment lifecycle. The common API overcomes the variety and heterogeneity of the technologies, languages and protocols used at the testbed level. SFA, OMF/FRCP

[51] http://fiteagle.org

and OML are the de-facto standard protocols or APIs that are used in Fed4FIRE federation to support the experiment lifecycle.

From the viewpoint of the federation operation and stakeholders' interaction, the model followed by Fed4FIRE is similar to the one-stop-shop [77]. The federator provides all the support to find and acquire the right to use the infrastructures' resources, i.e. resource discovery and reservation. The experimenter invokes services on the infrastructures directly. Furthermore, as with the matchmaker model, there might be a need for contracts or SLAs between the federator and the federation members, so as to determine what the federator can and cannot do [77].

2.5. Monitoring Concepts and Solutions

Monitoring is an essential part of every ICT infrastructure. It facilitates the control and management of different entities in various application areas, such as computing, storage and network resources, networking and connectivity, systems, processes, applications, and traffic flows. As per its definition (see Sec. 1.1), monitoring is the process of observing and recording information about these entities to determine their state, usage, performance, and behavior. This information is utilized by service components to make intelligent control and management decisions, e.g. for health and performance assurance, capacity and resource planning, security and privacy assurance, SLA management, data analytics, etc. Furthermore, monitoring is also required by users who are interested in obtaining information or even executing the measurements themselves to determine and evaluate their services or applications.

As the scope of this thesis includes the federation of cloud and FI experimentation facilities, the focus in the following will be on the monitoring aspects within these areas. Thus the main stakeholders are the infrastructure providers, federation services (provided by the federator) and users (experimenters in experimentation facilities and cloud services or application providers in cloud facilities). Infrastructure providers are looking for rich monitoring information of their resources (physical and virtual) to ensure their operational health and availability, as well as to understand the performance and the behaviors of the deployed resources, services and platforms for effective and efficient decision-making in terms of provisioning, management and optimization. Information is required by federators to manage federation-wide services, such as keeping track of the health and availability of infrastructures and providing knowledge for all parties in the federation through a FLS dashboard, SLA management, brokerage and reservation, charging and billing, etc. Users are interested in monitoring information about the used resources and are sometimes looking for capabilities to execute measurements themselves to determine and evaluate their services or applications. In experimentation facilities, the users (referred to as experimenters) are the major monitoring stakeholders, as measuring experiment

resources and collecting observations are an essential part of any scientific evaluation or comparison of technologies, services or applications being studied.

2.5.1. Monitoring Concepts

This section gives an overview of different concepts and terms related to monitoring discussed throughout the thesis.

2.5.1.1. Monitoring Process Stages

Usually, the monitoring process goes through multiple stages. It starts with producing the information from particular Measurement Points (MPs), considered to be data sources, using different kind of tools (capturing tools, measurement probes, agents or clients), depending on the application area, for instance, utilizing the Iperf[52] tool to get delay information between two points. This step is also referred to as data acquisition. After that, the data might be transported to an external collection resource. The data might be processed, i.e. filtered or sampled (usually this happens before being transported to external collectors), and then stored or directly provided for users if data storage is not required or supported. If it is stored, for instance in a database, it can be accessed, visualized or analyzed. In some cases, where needed, the data can be archived for post-processing and future use.

As already mentioned, measurements can be performed as part of the whole monitoring lifecycle in order to acquire monitoring information. Measurements can also be executed as a stand-alone process in order to check particular states or values at any point in time. The definition of measurement used within this thesis is as follows.

Definition of Measurement: As defined by Weiner [87] measurement is: "A systematic, replicable process by which objects or events are quantified and/or classified with respect to a particular dimension. This is usually achieved by the assignment of numerical values."

Measurements can be performed in any environment to monitor either hardware or software. Measurement data can also be obtained in different ways, by diverse tools, with data then utilized by various types of consumers. Within the context of this thesis, measurement data is acquired from various sources that expand vertically (*cross-layer*) and horizontally (*cross-domain*) according to points of observation.

2.5.1.2. Cross-Layer Monitoring

New models of offering services, driven by emerging technologies and paradigms like Cloud Computing, SDN and IoT, vary in their deployment and operation. To give an

[52] https://iperf.fr

example from the cloud domain where multiple stakeholders are present (infrastructure providers, cloud services providers and cloud services consumers), IaaS services are deployed and offered differently than PaaS and SaaS services. Their consumers also receive different control permissions, as discussed in Sec. 2.2.2. In such environments, stakeholders will have different interests in the information about their resources, which again varies from layer to layer depending on the type of service and the level of permitted control. *Cross-layer* monitoring services are of major interest from low-level hardware resources (e.g. CPU, memory), through to OSs, networks, platforms and up to the application layer [32], [64], [88], [89].

Definition of Cross-Layer Monitoring: Monitoring all functional layers expanding from low-level resources (e.g. CPU, memory) through the network up to the services and applications layers.

2.5.1.3. Cross-Domain Monitoring

The importance of obtaining monitoring information at multi-domain level (or across domains) is present in many ICT sectors. An example is the Wireless Mesh Networks (WMNs) domain, which was one of the inspiration that motivated the need for a monitoring solution operating across domains in this thesis. In [55], virtual, context-aware WMNs are built with the help of network virtualization to serve different service providers. This allows an end-user to have multi-access through a single access technology/device, thus, being able to connect to the best fit Virtual Network (VN). Monitoring is a fundamental task to support the management of multiple VNs, which may have different context characteristics (i.e. different QoS, mobility and security configurations). It is necessary to monitor the virtual resources, which form the VNs themselves, as well as QoS, mobility and security related metrics. In addition, the energy consumption is a further significant characteristic to be considered. In [90], different methods have been considered to save energy in wireless federated infrastructures. Virtualization is a key technology that enables saving energy in different ways, not only by virtualizing resources to create multiple software-based instances on top of one physical one, but also by virtualizing wireless access interfaces that can be used to build WMNs [56]. When detailed monitoring information is lacking in such meshed (federated) environments, it would not be a trivial task to optimally and efficiently manage their operation. Such information may include details of the infrastructure resources (e.g. energy consumption and resources availability) and the state of the load (e.g. the number of subscribers and their usage)

Beyond its importance in grid infrastructures [91], *cross-domain* monitoring support is essential in federated infrastructures within the context of Cloud and FI testbeds and experimentation [29], [37], [89], [92]–[94], as MPs are distributed across multiple domains that might be of different nature (wired, wireless, cellular, virtualized

infrastructures, etc.) and independently administrated, e.g. [11]–[13], [19]. In such environments, both *cross-layer* and *cross-domain* monitoring support is basically required, as different level of information across multiple, distributed domains is required by the various types of consumers. For example, infrastructure providers look for information on the health and performance of their resources; federation services are looking for information on the availability status of resources across the federation, SLA validation, interconnectivity across-domains; and users are looking for information on their usage and service quality (at the services and applications level), where services and applications resources are deployed across domains.

Definition of Cross-Domain Monitoring: Monitoring various components, systems, network and software metrics that are distributed across multiple domains.

2.5.2. State-of-the-Art Monitoring Solutions

This section gives an overview of some of major tools and solutions currently used by cloud infrastructures and FI testbeds. As OML and Zabbix[8] are the most relevant solutions in this thesis, they are discussed in more detail in separate sections.

2.5.2.1. Review of Monitoring Solutions in Cloud Computing

In commercial and experimental cloud infrastructures, there are many monitoring systems and solutions being used. Cloud monitoring solutions are mainly targeting homogeneous, single-entity administered cloud infrastructures [26], [27], [31]. Examples include Ganglia[53], Nagios[54], Zabbix[8], collectd[55], OpenNMS[56], Groundwork[57], CloudStatus[58], CA Unified Infrastructure Management[59] (formerly CA Nimsoft Monitor), MonALISA[95], mOSAIC[96] and CASViD[97].

Moreover, several monitoring architectures targeting cloud management and monitoring have been proposed in the literature. For instance, the service oriented monitoring solution [98]–[100] of the European OPTIMIS project [101] is one of the OPTIMIS Toolkit's software components. It is used to provide monitoring information about virtual and physical resources of cloud infrastructures and services in order to support self-management and optimization processes. It is based on open-source tools, such as the monitoring tool Nagios[54], which is extended through the implementation of NEB2REST (RESTful Event Brokering module) to interact with a RESTful Web service [102] for monitoring resources and services [98]. The private cloud monitoring

[53] http://ganglia.sourceforge.net
[54] http://nagios.org
[55] http://collectd.org
[56] http://www.opennms.org
[57] http://www.gwos.com
[58] http://cloudstatus.eu
[59] http://www.ca.com/us/opscenter/ca-unified-infrastructure-management.aspx

system (PCMONS) [103] is another solution that is based on Nagios[54] as well. The elastic monitoring framework introduced in [104] enables monitoring resources from low-level metrics from OSs to higher level application-specific metrics derived from services.

The aforementioned systems and architectures address the monitoring of cloud environments but cannot be applied to federated ones where a large number of resources from heterogeneous infrastructures are offered to customers.

However, the monitoring solution developed within the European project RESERVOIR [105] partially supports some cloud federation aspects. It provides information about services deployed in federated clouds for service management purposes. Yet, this solution does not provide information to cloud-service customers. In contrast, the Amazon monitoring system CloudWatch[60] provides monitoring data to customers regarding their running services, rather than providing data for infrastructure and service management.

These solutions have not been designed to serve different types of users (e.g. infrastructure management and cloud services customers). Furthermore, even though some of them might provide partial cross-layer monitoring information, they can't be deployed in a form of *Monitoring as a Service (MaaS)* so that each user can get an instance of the entire monitoring solution for exclusive use with full control.

This will be discussed in more detail in the requirements (Chapter 3), as well as in the initial design (Sec. 4.1.1) of the solution delivered by this thesis. Support is required for monitoring on network and infrastructure levels from heterogeneous domains, as well as providing monitoring support that run across large populations of end-to-end resources at the service and application levels.

2.5.2.2. Review of Monitoring Solutions in Future Internet Testbeds

Concerning monitoring solutions in FI testbeds and experimentation facilities, a wide range of tools and frameworks are used. Many testbeds develop and use their own tools or use third party open-source tools that are used for monitoring distributed systems, such as Zabbix[8], Nagios[54] and collectd[55]. However, the major ones are reviewed in [86] and briefly below.

INSTOOLS [106], [107] is a system of instrumentation tools that enables GENI users to monitor and understand the behavior of their experiments by automatically setting up and initializing experiment-specific network measurement and monitoring capabilities on behalf of users. In its current implementation, the measurement software is deployed along with the experiment resources to be installed on the MPs. The collector collects data and provides it to the user through a Web interface. INSTOOLS is limited to a set of measurements based on classical measurement tools,

[60] http://aws.amazon.com/cloudwatch/

such as Simple Network Management Protocol [218] (SNMP), tcpdump[61], Cisco IOS NetFlow[62] and standard OSs tools like `ps` and `vmstat`.

There are some efforts to support monitoring across federated infrastructures, but these are limited to specific domains or measurement services. For instance, the monitoring architecture of the Networking innovations Over Virtualized Infrastructures [35] (NOVI) research project [35] uses four monitoring tools deployed across heterogeneous virtualized infrastructures. It is limited to a specific use (traffic monitoring) in a specific domain (virtualized distributed server systems). These tools are 1) Hades Active Delay Evaluation System (HADES), used for one-way delay, loss statistics, and hop count information, 2) packet capturing cards for line speeds up to 10 GBit/sec, 3) the Multi-Hop Packet Tracking [211] (Packet Tracking), an efficient passive one-way delay measurement solution, and 2) Service Oriented Network Measurement Architecture [219] (SONoMA).

SONoMA is a Web Service based network measurement platform. It supports the most common network measurement types (e.g. bandwidth measurement, delay, network tomography, geographical localization) through a set of functions that are supported through a standardized SOA-based Web services interface. A user simply determines what needs to be measured and the system will perform the process. The data is then forwarded back to the user and automatically stored in the public data repository, called Network Measurement Virtual Observatory[63] [196] (nmVO).

Another solution to support monitoring over multiple infrastructures is Continuous Monitoring [176] (CoMo). CoMo provides a unified data interface but is limited to passive network measurement. The European Traffic Observatory Measurement Infrastructure [179], [180] (ETOMIC) is another network traffic measurement platform that is synchronized with Global Positioning System (GPS) and allows its users to infer network topology and discover its specific characteristics (e.g. delays and available bandwidths). Users are able to deploy their own active measurement codes to conduct experiments over the public Internet.

TopHat [108] is another solution that provides topology monitoring and network measurements and supports experimentation across federated testbeds. It uses a dedicated measurement infrastructure called TopHat Dedicated Measurement Infrastructure (TDMI) that runs active measurements, including Paris Traceroute. TopHat provides access to a set of interconnected systems allowing its users to get measurement information from them. Among those supported systems are DIMES[64] [178] (DIMES), ETOMIC, SONoMA, CoMo, MySlice[65] (MySlice) and Team Cymru[66] (Team Cymru).

[61] `http://www.tcpdump.org`
[62] `http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html`
[63] `http://nm.vo.elte.hu`
[64] `http://www.netdimes.org/new/`
[65] `http://myslice.info`
[66] `http://www.team-cymru.org`

Performance focused Service Oriented Network monitoring ARchitecture [38], [92] (PerfSONAR) is a multi-domain network performance monitoring framework that focuses on offering network performance characteristics (e.g. delay, packet loss, link utilization and interface error). It has been developed by the Internet2[67], Energy Sciences Network[68] (ESNet) and GÉANT[69] in order to provide network administrators cross-domain performance information in an easy manner. The framework is divided into three main layers with different modules. The lower layer includes passive and active measurements performed over MPs; there is one module for each measurement metric (e.g. one for delay and another one for throughput). A second layer coordinates the management of measurement tasks with particular Web services. A third layer at the user level provides a user interface for data representation and visualization. PerfSONAR services each use a different tool for performing measurements. For instance, Iperf[52] is used by the Bandwidth Test Controller[70] (BWCTL) service for conducting throughput measurements, while the One-Way Ping[71] tool that implements the One-Way Active Measurement Protocol [201] (OWAMP) is used to retrieve one-way delay, jitter and packet loss measurements.

Leveraging and Abstracting Measurements with PerfSONAR[72] (LAMP) is a GENI project that creates an instrumentation and measurement system based on PerfSONAR in order to be used by experimenters on ProtoGENI.

Although these solutions can be run on multi-domain environments, they are limited in their support to specific fields of applications, in particular network measurements. Extending any of these to support cross-layer and cross-heterogeneous-domain monitoring services is costly in terms of time and effort, as well as creating scalability issues should new fields of application be covered. Still, there is demand for a generic and extensible solution capable of supporting various sets of monitoring services in federated, heterogeneous environment [11], [17], [28]–[31].

2.5.2.3. ORBIT Measurement Library Framework (OML)

OML is a distributed software framework enabling real-time collection of data in a large distributed environment. It is mainly used as the measurement part of OMF-based testbeds as a way to collect and process data from distributed applications and experiments, but can also be used as a standalone data collection and reporting system.

From architectural viewpoint, OML consists of two main components for injection and collection, and a third optional component:

[67] http://www.internet2.edu
[68] https://www.es.net
[69] http://www.geant.net/Pages/default.aspx
[70] http://software.internet2.edu/bwctl/
[71] http://software.internet2.edu/owamp/index.html
[72] http://groups.geni.net/geni/wiki/LAMP

- **OML Client Libraries** There are multiple client libraries written (and supporting software) in different languages: C-library (*liboml2*), Python (*OML4Py*), Ruby (*OML4R*), Java (*OML4J*) and JavaScript (*OML4JS*). The C library is native to OML, which provides an API (written in C) for applications or any software. It facilitates both collection of measurement data and its encapsulation as OML streams sent to one or more local or remote collection endpoints, called OML servers, using OML Measurement Stream Protocol[73] (OMSP).

- **OML Server** It receives measurements collected from distributed applications and stores them in a database backend (SQLite3 and PostgreSQL are natively supported). OML server is able to collect data from multiple domains at the same time. Domains can be identified through a given experimental domain name and a sender Identifier (ID).

- **OML Proxy Server (optional)** In case OML is used to report data in an environment involving disconnections from the control network (e.g. in the case of reporting data from mobile devices), the proxy server can be used to temporarily buffer data until a connection is available to transfer the buffered data to the collection server.

OML supports two methods to produce measurements:

- OML allows application developers to define customizable MPs embedded in new or pre-existing application code. Users running the applications can then direct measurement streams from these MPs to OML servers. OML includes a program called *Scaffold*, which facilitates generating MPs of the instrumented application based on configurable templates.

- For applications for which code is not available, it is still possible to collect data provided by them and report such data to the server through writing simple wrappers using one of the supported languages (e.g. C, Python, Ruby, Java, or JavaScript).

One the of the significant advantages of OML is its ability to allow the creation of flexible, user-defined schemas of the streams carrying the data. Such a feature makes OML a generic solution that can be used in various application areas for data reporting. However, such flexibility could lead to interoperability and semantic problems if data is reported by diverse tools or users, as it still lacks common, widely used, standardized vocabularies or information models [109].

[73] http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html

2.5.2.4. Zabbix Monitoring System

Zabbix[8] is an open-source, distributed monitoring solution. It follows the client-server model, and thus, consists of two major software components: server and agent. Zabbix allows monitoring of a wide range of devices and software components, such as servers, network devices, databases and applications. It supports performance monitoring natively, in addition to an extensive list of OSs and platforms. Besides the built-in checks (provided by default), it is possible to create customizable, user-defined checks on agents. This facilitates automatic and periodic application monitoring.

Zabbix supports different possibilities for data acquisition: through the use of native Zabbix agents (in active or passive mode), SNMP agents or agentless script-based queries (checking the availability and responsiveness of standard services like mail or Web servers without the need to install any software on the host devices). In the use of passive agent mode, the server asks for some data that is then sent back by the concerned agents. In active mode, agents report data to the server. Agents are small software components that are configured to send measurement data about predefined metrics to the server at regular intervals. Agents typically produce the data by executing Unix scripts written to obtain the measurements. Data is then collected in a central collection server that includes as a storage backend one of the supported Structured Query Language (SQL) databases (MySQL, PostgreSQL, Oracle or Microsoft SQL Server).

Besides a Web interface, Zabbix exposes a JSON-RPC-based API allowing users to read, visualize and retrieve data in addition to performing Create-Read-Update-Delete (CRUD) operations for various relevant objects: metrics (called items in Zabbix), templates, graphs, aggregated and calculated information, and triggers. In addition to problem detection based on predefined triggers, Zabbix can send email notifications or report Short Message Services (SMSs) to notify administrators about a current or potential problem, or even execute shell scripts or any other user-defined actions.

Depending on the number of checks, Zabbix can scale up to tens of thousands of servers, VMs and network devices that can be monitored simultaneously in real-time. In such large environments, an automated way of monitoring devices is required. Zabbix, through its automated registration and discovery features, supports automated setup, configuration and management. Thus, monitored devices can be added and removed automatically as they come and go. However, Zabbix has its own housekeeping process that is executed by the server periodically to remove outdated data. Monitoring data of a measured item is stored for a specific period of time (given in days). This history property of each item, initially is set to a default value, can be manually changed on demand based on, for example, the available storage size, or the time interval during which the data is required.

2.6. Data Modeling

This section presents state-of-the-art approaches to information and data modeling. Before describing such approaches, related concepts are first defined. An important distinction is that between information and data models. A data model, compared to an information model, can be seen as a lower level of abstraction that includes more details [110]. In addition, a data model is represented in formal data definition languages specific to the used protocol. From this viewpoint, as conceptual models can be implemented in different ways, multiple data models can implement a single information model.

Definition of Information Model: An information model is used by designers to model entities and their relationships within a given domain at a conceptual level [110].

Definition of Data Model: As defined by NOVI, a data model "describes protocols and implementation details, based on the representation of concepts and their relations provided by the information model".

Various challenges are faced while analyzing the possibilities for defining a common monitoring information model for complex, federated ICT environments. This is because of the diversity and heterogeneity of the domains involved, tools used, resources (virtualized and real) offered, application areas, and use and access policies. Despite these challenges, there are a number of different approaches to developing information and data models, as presented in the following sections.

2.6.1. Integration of Heterogeneous Databases

Integration of heterogeneous databases is one possible method to have unified access to heterogeneous data resources. This type of integration is not a simple task to achieve due to the various data structures and semantics of the integrated schemas. Therefore, many methods for databases integration exist in the literature. They vary from one another depending on the associated requirements and usage. Some methods are based on the use of mediated, global schemas at integration level and local wrappers above the databases and, thus, allow users to have transparent access to data [111]. Others are based on the use of multi-layered schema architectures (ranging from local schemas, export schemas and import schemas to global schemas), where each layer presents an integrated view of the concepts that characterize the layer below [112], [113].

This approach assumes the definition of a particular architecture design for data access. Such a solution does not scale well in the case of the use of diverse sets of data sources, nor it is flexible and generic enough to accommodate new data sources

easily. Furthermore, this approach lacks semantic information, expressiveness and interoperability of data exchange between components. As such, this approach is not seen as a valid approach to fulfill the requirements of the solution delivered by this thesis, which aims to provide a flexible, scalable and generic solution for the common representation of data collected and stored by arbitrary monitoring and measurement tools and frameworks.

2.6.2. Information and Data Models

There are multiple information and data modeling solutions for common monitoring data exchange, but they are limited to particular domains and/or are designed in a task specific manner.

For instance, the OGF Network Measurement Working Group (NM-WG) worked to provide shared knowledge about measurement tools and metrics [114]. NM-WG developed an infrastructure that allows the exchange of network measurement data and knowledge between different systems. NM-WG mainly focused on the definition of a common vocabulary used to provide information about and from different measuring tools. For each of the considered tools, they defined a particular kind of Extensible Markup Language (XML) schema [115], called RELAX NG (Regular Language for XML Next Generation). Information is then sent in XML code defined by that schema. Another example is the information representation schema of PerfSONAR. It is based on the NM-WG's XML schema that is used to represent network performance monitoring data.

According to [116], these solutions have two drawbacks:

1. they are based on an XML schema that supports only a common syntax but does not allow deducing any information from monitoring data;

2. they are limited to a set of services, i.e. they don't cover every common monitoring concept that exists in large-scale federated, heterogeneous infrastructures.

Furthermore, there are other data models, such as those of the Internet Engineering Task Force (IETF) SNMP and Internet Protocol Flow Information Export [120] (IPFIX)[117] protocols, which are discussed in the following section.

2.6.3. Data Transport Protocols

Various protocols are used by the state-of-the-art solutions to transfer monitoring data between agents and collectors. In this section, those solutions and the associated protocols adopted in FI testbeds are discussed.

Some solutions are equipped to deal with the variety of tools, data formats and functionalities, such as DIMES, Measurement Infrastructure for Network Research [191] (MINER), TopHat [108], PerfSONAR, INSTOOLS [106], [107] and more, as

discussed in [118]. These solution aim to abstract the different tools using a common way of manipulating them through APIs without providing much detail on how the data is transmitted, limiting description in some cases to a mention of the use of Transmission Control Protocol (TCP) streams or capability to support standard protocols like SNMP and ICMP as it is the case for PerfSONAR and INSTOOLS.

Other solutions, such as SNMP, IPFIX and OMSP, are standard or de-facto-standard protocols that are used for transmitting data records at application level. They still rely on the traditional transport protocols, i.e. TCP and User Datagram Protocol (UDP). These three are discussed in detail as follows.

2.6.3.1. Simple Network Management Protocol (SNMP)

SNMP is an application layer protocol defined by IETF. Its main goal was to have a simple means of accessing and managing IP enabled devices remotely. Its usage has become wider and is often used for network monitoring and fault management. SNMP is a bidirectional protocol and is based on agent-manager architecture. It supports two bidirectional data exchange. First, the manager sends queries to the agent, who responds by sending the desired information. Second, the agent reports information once a specific, pre-defined condition is met.

The relevant part of the standard to this thesis is the type of information that can be queried, as well as the data structure and format. This is defined by a set of managed objects called Management Information Base (MIB), which aims at abstracting the features and behavior of the agent in a sort of database. MIB should be known and agreed between both the manager and the agent. SNMP is constrained to report only information predefined in its MIB. Although SNMP is an extensible protocol, which can be extended to allow the transmission of heterogeneous data, it needs pre-configuration of both the manager and agent each time new sets or even new structure of data needs to be reported. In addition, SNMP does not scale well as evaluated in [119]. It is restricted to counter or database fields. Scenarios requiring flows of information could not be supported by such a protocol. SNMP is, however, a standardized protocol and is supported by a large range of devices and tools. It is therefore a consolidated protocol that should be taken into consideration when deciding on the best alternative to transport measurement data.

2.6.3.2. Internet Protocol Flow Information Export (IPFIX)

Similar to SNMP, IPFIX [117] is an application protocol designed by the IETF, however it is developed for exporting Internet Protocol (IP) traffic flow information over the network. Its basic function is to transmit a flow information in a unidirectional manner from an exporter to a collector. An extension to the protocol was introduced in [120] in order to cover every kind of measurement through the so-called template

approach, which gives flexibility concerning data format. Furthermore, IPFIX has already been applied in completely different scenarios from the ones it was originally designed for, e.g. in wireless sensor networks [121].

IPFIX can use any transport protocol. However, it is preferred to be implemented over Stream Control Transmission Protocol (SCTP), while TCP and UDP can be used. SCTP is preferable due to its reliability and security support, as it is congestion aware and handles multiple connections.

Datasets and their schemas are defined through a template that is submitted to the collector from an exporter prior to sending data. A template contains a schema of the data coming next, which will include the template key beside the length of the packet as overhead fields. The literature lacks reporting on the performance of IPFIX, however, according to NetQoS [122], IPFIX typically increases CPU utilization on the configured devices by only 1% to 2% on average. The bandwidth consumption depends heavily on what kind of flow is being exported and the transport protocol used. The additional bandwidth caused by the header of the protocol should not be of concern in most scenarios.

2.6.3.3. OML Measurement Stream Protocol (OMSP)

Unlike SNMP and IPFIX, OMSP is not standardized yet. However, it is currently being used as part of the OML framework [123], which is used in several FI testbeds. OML is based on a client-server architecture. Similar to IPFIX, OMSP is unidirectional and follows the concept of clients submitting data schemas to the server before sending the data itself. Data can be then sent either in text or binary format.

Concerning the transport protocol, OML always uses TCP streams. A client opens a TCP connection with an OML server. After that, the client sends the schema, followed by the measurement tuples. OML allows collection and representation of any type of measurement data from any distributed applications in a unified format.

Similar to IPFIX, within one connection, OMSP client first sends data schemas in a header, followed by the data. These schemas are stored in the OML server. These schemas can be updated or additional ones can be added at any time. A significant difference to IPFIX is that OMSP is not standardized and is therefore limited to a single implementation. Nevertheless, OMSP is capable of carrying any type of data supporting heterogeneous measurements reported by any application from any domain. This significant capability makes OMSP the most appropriate candidate for transporting arbitrary measurement data with unlimited freedom across distributed measurement systems in a federation environment.

2.6.3.4. Summary

One of the basic requirements of a distributed monitoring solution for a federated infrastructure is that its data reporting protocol should be scalable, flexible and generic enough to carry various sets of measurement data. It should also be understood by other applications at the collection endpoints in order to be able to correctly interpret diverse datasets.

The traditional, standardized protocols used by the state-of-the-art frameworks can't be adopted without further modification to support this requirement as discussed in the previous section. Although SNMP and IPFIX support unified data reporting, they are limited to particular domains (i.e. network measurement). SNMP only reports information predefined in its MIB. IPFIX leverages SNMP's MIB and has limited representable information focusing on IP traffic measurement.

In contrast, the OML framework uses OMSP, which is a very flexible protocol, able to carry heterogeneous datasets following pre-defined schemas transmitted as streams. OML is not limited to particular domains but is generic enough to instrument the whole software stack (from low-level resources through networks up to applications), and even take input from any software interface. It has no restrictions on the type of software to be instrumented, nor does it force a specific data schema. Rather, it supports whichever schemas or streams are in use.

The adoption of the OML framework with its OMSP protocol as a common interface across the federation allows data to be provided in a common format. However, this solves part of the problem, as it allows data to be represented in only flexible, user-defined schemas, but not in a meaningful, standardized way. There is a need for common information and data models to represent monitoring concepts and relationships in unified manner following a shared vocabulary [109].

2.6.4. Ontology-Based Modeling

Ontologies are used to describe real world things in a formal and explicit vocabulary. They are widely used in the Semantic Web [216] (Semantic Web) to describe Web services. However, they have also gianed significant attention recently in other domains, such as BigData, sensor networks and IoT.

Definition of Ontology: "An Ontology is a formal, explicit specification of a shared conceptualization. [124]" An ontology defines a set of formal, explicit vocabularies and definitions of concepts and their relationships within a given domain. Concepts can be classified semantically (in terms of their meanings). This is achieved through classes, subclasses and their instances.

According to [125], "Ontologies are content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified

domain of knowledge. They provide potential terms for describing our knowledge about the domain." They allow common vocabulary and knowledge to be defined and shared between multiple entities. Such a common definition of knowledge, facilitates data exchange, merge and combination among different components. Unambiguous names of terms allows data to be bound to real world objects. They also allow data reasoning through logical rules.

The following briefly describes the main ontology components used to develop an information model:

- **Classes** Classes refer to concepts. A class represents a group of **individuals** that share common characteristics, also called instances of the class. Multiple classes can be declared as subclasses of the same superclass.

- **Relations** Relations describe relationships between different concepts (classes and subclasses), e.g. *subClassOf*. Concepts usually have **properties** that can represent relations. Each property has a **domain** and **range**. The domain links a property to a class or an individual. The range links a property to a class, an individual or a data range (attribute).

- **Attributes** These describe parameters, features or characteristics of a class. They are represented as **data properties**.

- **Axioms** These represent assertions (including inference rules) allowing the insertion of new facts and predicates.

There are also further components including the following:

- **Annotation Properties** These allow annotations (e.g. labels, comments) on classes, properties, and individuals.

- **Restrictions** Restrictions can be value constraints "on the range of the property when applied to particular classes" or cardinality constraints "on the number of values a property can take" [126].

Ontology-based information models are extensible to include additional vocabularies. Among the main strengths of ontology-based modeling are that it is standardized and expressive, it focuses on interoperability, schemas are unbound and it is extensible [127]. Information models described through ontologies could be represented by any of several possible languages [128] such as Frame Logic (F-Logic), Ontolingua, Knowledge Interchange Format (KIF), LOOM, and Web Ontology Language [126] (OWL). OWL is the most widely used and powerful language for representing knowledge (publishing and sharing ontologies) in the Web.

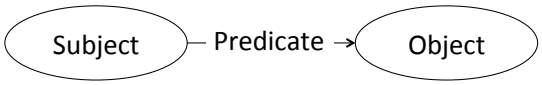


Fig. 2.6.: RDF graph

Ontologies use the general-purpose language Resource Description Framework [213] (RDF) as a data model, which allows describing and organizing the data as triples of the form subject-predicate-object in graph format, as shown in Figure 2.6.

Figure 2.7 illustrates an example of how information is represented in a graphical format, where any measurement metric has measurement data and is measured by a measurement tool.

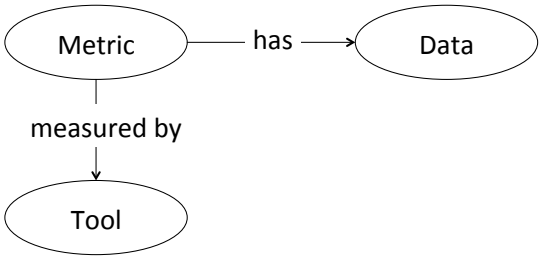


Fig. 2.7.: Information represented in RDF graphical format

RDF uses different formats for serializing (expressing) RDF triples. RDF/XML Syntax[74] is the first standard RDF serialization format that serializes the triples as an XML document. Other alternative serialization formats are RDF-Turtle[75], which allows writing graphs in a natural text format; RDF N-Triples[76], which enables a line-based, plain text format; and Notation3 (N3)[77], which is a readable RDF format. RDF provides basic vocabulary that is then extended in Resource Description Framework Schema [212] (RDFS). RDFS adds data-modeling vocabulary for RDF. OWL is built on and extends RDF and RDFS with additional vocabulary and semantics. These can together be seen as a framework for modeling (describing) all forms of data through vocabulary and for allowing data interoperability through shared conceptualizations and well-defined, standardized schemas.

Figure 2.8 shows how an information model can be defined through an ontology in e.g. OML, which can then use any suitable data model (e.g. RDF) that in return uses any of the possible serialization languages for expressing the data.

Finally, it is worth mentioning that the exchange of information in an ontology-based way is only valid across tools and services that are aware of the ontologies in question.

[74] http://www.w3.org/TR/rdf-syntax-grammar/
[75] http://www.w3.org/TR/turtle/
[76] http://www.w3.org/TR/n-triples/
[77] http://www.w3.org/TeamSubmission/n3/

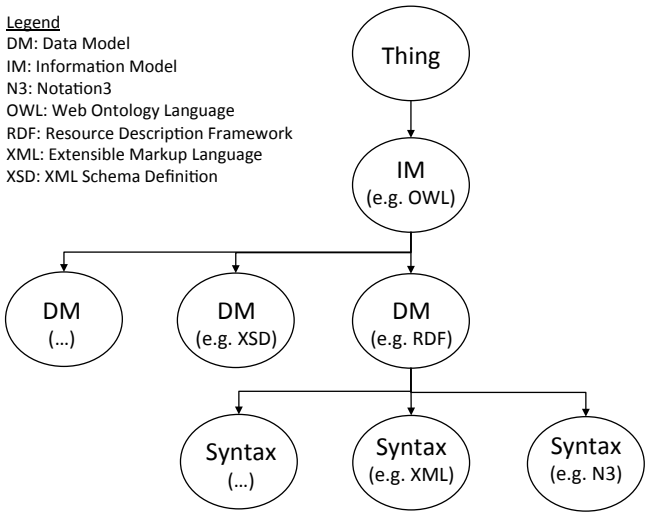


Fig. 2.8.: Relationship between information model, data model and syntax (based on [129])

2.6.5. Applied Ontologies

Ontologies have been applied in several fields. Semantic annotations have been devolved for the Web Services Description Language (WSDL) and XML schema under the so-called Semantic Annotations for WSDL and XML Schema [214] (SAWSDL). Search engines including Bing, Google, Yahoo! and Yandex have collaborated to create the shared markup vocabulary called Schema.org[78]. It provides a shared collection of schemas to annotate websites, allowing search engines to recognize them easily, improve the display of the search results, and facilitate structured data interoperability. Semantics have been also introduced for the IoT domain in [130].

Ontologies have been introduced in the converged telecommunication networks in order to solve interoperability issues, caused by the use of diverse information and data models [131]. Barcelos et al. have studied the use of ontologies to assist some of the ITU-T recommendations [132]. This is because the ITU-T models, beside being limited to the management of the telecommunication networks [133], are written in natural language to be understood by human. Furthermore, these models contain many ambiguous declarations that lead even to human misunderstandings in their interpretation [132].

In the fields related to this thesis, there are a number of relevant ontologies. The Network Mark-Up Language [195] (NML) is an information model designed to describe computer networks. NML, an OGF standard, was designed to be as general as possible to be easily extensible to cover further emerging network architectures.

[78] http://schema.org

Another example, the Infrastructure and Network Description Language [134] (INDL) semantically modeled computing infrastructures [134].

There are a number of ontologies focusing on monitoring. Some are out of the scope of this thesis, such as MonONTO [135], which focuses on advanced Internet applications and QoS related concepts, and the Semantic Sensor Network [136] (SSN) ontology [136], which focuses on the domain of sensor networks. The following ontologies are highly relevant and even partially reused in the ontology developed within this work.

The aim of the Monitoring and Measurement in the Next generation Technologies [34] (MOMENT) ontology [34] is to provide a unified interface to harmonize network measurement data produced by heterogeneous sources. It builds upon state-of-the-art solutions and information models in the domain of network measurements, like the measurement data standardization of NM-WG, the most common external data representation from IETF, Applied Internet (CAIDA) Internet Measurement Data Catalog (DatCat) [137] and data accessing possibilities provided by PerfSONAR.

NOVI information models [35] include three ontologies describing resources, policies, monitoring capabilities and communications in the NOVI architecture, which focuses on the federation of virtualized infrastructures. The design of the monitoring ontology is based on the requirements of NOVI architecture and therefore includes seven models that describe seven main monitoring functionalities (Query, Task, Event, Feature, Parameter, Unit and Statistics).

The MOMENT and NOVI monitoring ontologies contributed to the Monitoring Ontology for IP Traffic Measurement (MOI) [36].

2.7. Summary

This chapter presented the state-of-the-art technologies, methods, and solutions related to the work conducted in this thesis. It covered technologies and paradigms such as service orientation (Sec. 2.1), Cloud Computing (Sec. 2.2), and virtualization (Sec. 2.2.4) as well as resource and infrastructure federation models and approaches (Sec. 2.4).

Sec. 2.3 gave a detailed overview of the major FI experimentation facilities and testbeds relevant to this thesis. These facilities are used to study and evaluate technologies, services and networking paradigms of the future.

Furthermore, monitoring concepts and solutions were discussed in Sec. 2.5. The limitations of the state-of-the-art solutions were briefly discussed.

Finally, as information modeling is at the heart of the work of this thesis, state-of-the-art models were discussed in Sec. 2.6. More emphasis was given to those based on Semantic Web ontologies, as those implemented within this thesis are ontology based.

3 Requirements Analysis

THIS chapter discusses the most relevant functional and non-functional requirements on monitoring federated ICT infrastructures. It includes a thorough analysis of the monitoring requirements of federated infrastructures. This analysis focuses mainly on Cloud and FI testbed infrastructures, which are used in this thesis as use cases for federation.

Furthermore, requirements of the emerging technologies and communication paradigms involved in such environments are taken into consideration. This analysis is conducted after discussing the state-of-the-art technologies, tools, standards, commercial solutions, international projects and research activities on aspects related to monitoring and federated Cloud and FI testbed infrastructures in Chapter 2.

It is worth mentioning again that the work in this thesis doesn't focus on monitoring specific domains, such as cloud, wireless or cellular, nor a particular level of application, such as the hardware, the network or the application level. Instead, its main focus is

on monitoring large-scale, distributed and federated environments, taking into account the demands and features of the heterogeneous domains involved.

As a matter of fact, this thesis will not deliver new tools or solutions for monitoring particular domains, since a wide range of solutions are available on the the market and in the literature. For instance, for monitoring network performance, a range of tools exist that differ from one another in their functionalities. As already mentioned in Chapter 1, this thesis aims at delivering an architecture that operates across federated infrastructures supporting monitoring at different levels. This can be achieved through the re-use, extension, adoption and adaption of suitable tools amongst the state-of-the-art ones and the addition of new features or services that are missing in these tools.

In this chapter, a set of requirements collected from different relevant sources are first explored. Thereafter, a through requirements analysis is conducted followed with a gap analysis.

3.1. Sources of Requirements

This section gives an overview of the general monitoring requirements in the concerned domains addressed in this thesis. It considers technical and non-technical requirements of multiple stakeholders interacting within these domains. Of course, these requirements are limited to those required

- to support and cope with the needs of emerging technologies, models and networking approaches (e.g. Cloud Computing, SDN, NFV, IoT),
- for the management and operation of the FI experimental facilities and the federation thereof, and
- by the users to check the performance and behavior of their deployed services or applications.

3.1.1. Emerging ICT Technologies and Paradigms

In recent years, several ICT technologies and computing and networking paradigms have emerged. Examples that are relevant to this thesis include Cloud Computing, SDN, NFV and M2M/IoT.

The Cloud Computing business model has changed the way ICT services and applications are being offered and managed (Sec. 2.2). For rapid use and adoption in many sectors, suitable monitoring methods and tools are required to support its functionalities [88], [138]. Amongst those tasks that require monitoring support as discussed in [88] are datacenter infrastructure resources management, capacity and resource planning, SLA, performance and security management, billing and

troubleshooting. The monitoring information assists cloud providers and application developers (cloud services users) as follows [17], [64], [138]: to keep resources and applications (e.g. PaaS, IaaS) operating at peak efficiency; to detect variations in their performance; to manage SLAs; to track the changes of resource deployment to cope with the elastic, dynamic and rapid way of resource creation and termination [139]; and to continuously monitor resources in order to timely predict possible failures (e.g. failure could happen in an overloaded server) that can be proactively handled [27]. Cross-layer monitoring support in different granularities is required to monitor the entire cloud stack [26], [88], [138]. This includes the ability for users to define own customizable metrics to measure their services or applications [27].

SDN is another example that has tremendous potential as a networking paradigm, working to facilitate programable control and orchestration of network services. SDN is seen as a complementary approach to NFV, where it facilities the orchestration of the the Virtualized Network Functions (VNFs) [140]. This serves to keep up with the scale and fluidity of cloud resource configurations and provides the ability to control and dynamically change network behavior, thus allowing the placement of the VNFs where and when they need to be, and facilitating operation and maintenance procedures [141]. For controlling such dynamically changeable environments, a suitable monitoring solution is required. It has to be capable of auto-detecting and monitoring a large number of virtual resources belonging to different deployments, as well as adjusting on-the-fly to their changes [140].

SDN is also recognized as a promising paradigm for organizing isolated networks, as might be the case in dynamic WMNs [55], [142], where dynamic traffic engineering support can be gained through SDN [142]. As already discussed in Sec. 2.5.1.3, information about the WMNs' resources (virtual and physical, wired and wireless) is required; examples include utilization (e.g. CPU-, disk- and memory-usage and energy consumption) of compute resources at the backbone of the WMNs infrastructure, as well as link and network quality (QoS parameters) and load. Such WMNs are discussed here as they are built in a federation-like manner, where access networks (e.g. WiFi, cellular or sensor networks) might be separately administrated. Furthermore, they could be considered as an enabler for IoT [55], [142].

3.1.2. Testbed Management and Operation

Monitoring is a key service in any ICT infrastructure to facilitate its management and operation [86]. Information about the availability, performance, utilization and behavior of infrastructure resources, services and users are required for many purposes, like accounting, evaluation, validation and optimization. Multiple stakeholders are interested in monitoring [10], [19], [94], [107], [143]. Their interests differ from testbed to testbed depending on the nature of the testbed and the services offered by each.

Furthermore, requirements differ from one stakeholder to another depending on the type, granularity, frequency and delivery methods of the monitoring information.

According to input and feedback collected from several testbed operators within the context of the FIRE (Sec. 2.3.2) and FI-PPP (Sec. 2.3.3) initiatives, monitoring plays a key role in the management and operation of testbeds. From the testbed administrator's viewpoint, providing rich monitoring information is essential for effective and efficient decision-making in terms of provisioning, management and optimization. This is due to the fact that it increases understanding of the performance and behaviors of deployed resources, services and platforms.

Monitoring information is required during the entire lifecycle of the deployed service (or experiment), starting from the negotiation (SLAs) and reservation through to the provisioning, execution, run-time, and service termination. Furthermore, there is a need for providing collection of measurements required to extend the SLA understanding towards the fulfillment of the specific services requested.

Monitoring and measurements have even higher significance in testbed infrastructures, where, in addition to serving testbed administrator's needs, making observations is essential in experimental scientific research. Therefore, from the experimenter's viewpoint, measuring experiment resources and their environmental conditions and collecting observations are an essential part of any scientific evaluation or comparison of technologies or services being studied [10], [17], [19], [64], [94], [107], [143]. Amongst the functional requirements identified by the Fed4FIRE experimental facility for their users (experimenters) is the need to open up the state of the offered resources to all users. This allows users to choose the best resources for their experiments. A further requirement is to provide monitoring information about users' resources. Furthermore, users should be able to request on-demand monitoring and measurements services. Finally, users' data should be provided and managed in a secure manner.

3.1.3. Federation Operation

Monitoring information plays a major role in the management of any federation, its operation and services. Specific information is required at the federation level by several services. Information is required to check the availability status of infrastructures and their key services (FLS) [10], [11], allow federation management systems to ensure end-to-end network and service performance and the interconnectivity [144], ensure that SLAs are met (SLA management) [11], [145], and construct a quantitative view of the trustworthiness of each facility (trustworthy reputation) [11].

However, concrete functional and non-functional requirements need to be analyzed and identified. Those requirements needed to support the functionalities of a set of federation services are discussed first. This is followed by a discussion of the requirements commonly found among federations.

3.1.3.1. Monitoring Support for Federation Services

Federation architectures usually include a set of services that facilitate management [10], [70], [145]. These services can be offered by one or multiple providers in a central or distributed manner. Functionality of services relies mainly on monitoring information as discussed in the following.

First Level Support FLS is one of the central federation services and provides first-line support for users through high-level status information on testbeds and resources availability. This service displays the overall status of each testbed as a Red, Amber, and Green (RAG) status through a central FLS monitoring dashboard. Testbed status is calculated based on the availability of key components of that testbed. Information for FLS needs to be gathered from the individual testbeds periodically. A requirement of FLS is the ability to drill down from the testbed RAG status to see which components are degraded or down. Such monitoring information is also used by federation administrators or FLS operators to check performance and take proper action in case of issues in a testbed, e.g. notify the administrator of that testbed.

Service Level Agreement (SLA) Management SLA management requires monitoring information about the used resources on a per-service-basis in order to ensure that the agreements are met. Predefined metrics are agreed on between testbed providers and the SLA management. The data needs to be provided on a regular-basis during the service lifetime.

Trustworthy Reputation Similar to the SLA service, the reputation service requires monitoring information about the used resources on a per-service basis. Monitoring data needs to be provided by the testbed providers. It is compared with feedback from users provided as answers to questionnaires at the end of their services. Both inputs are used to generate the reputation score for each resource or service, with the score updated after every evaluation. Predefined metrics are agreed upon between testbed providers and reputation operators. An example from a wireless testbed could be channel interference, used to evaluate users' experience of interference during their services.

Reservation Brokerage This service allows users to create resources based on a set of selection criteria (e.g. time, type). The reservation broker requires monitoring information of physical resources from all testbeds over a predefined time window in order to efficiently map the requested resources to the capacity and availability of resources at testbed level. Such information is considered necessary to schedule (perform advance reservation of) resources in an intelligent and efficient manner. Required monitoring metrics are to be agreed on between the testbed providers and

the reservation broker operators. Examples from a cloud testbed include the number of free resources and the CPU-, disk- and memory-usage of physical machines.

3.1.3.2. Monitoring Challenges Across the Federation

To consider proper requirements of a sustainable federation, those of three practical, large-scale testbed federations are analyzed, namely BonFIRE, FIWARE Lab and Fed4FIRE. The Fed4FIRE federation, for instance, includes multiple heterogeneous FI testbeds that are distributed in various European countries, with each already serving numerous users. In such complex, large-scale, heterogeneous and geographically distributed environment, used by a large number of users, various sets of functional and non-functional requirements are identified, among these are the following:

- Users have different interests in monitoring services depending on the types of experiments planned. Therefore various monitoring services are offered in a flexible way (different configuration is possible), so that users have the ability to configure their requests on-demand.
- Numerous resources are deployed and monitored and large amounts of measurement data are collected from multiple testbeds and then provided to users. The proposed monitoring solution would consume resources (e.g processing resources, bandwidth and storages) for performing its services. These services include the execution of measurements, data collection, conversion into a unified format, transportation, and making data available to users. It therefore has to be designed in a way to minimize the consumed resources for supporting monitoring services and to minimize the impact on the experiments themselves.
- Data is collected from heterogeneous data sources, which might provide the data in various schemas and formats. Therefore, there is a need for harmonizing measurement data to be provided in a unified data representation and in a standard manner.
- Experiment resources may be deployed across the federation. It should be possible for a user to have data collected in a common way in one location.
- The user should be able to identify any collection endpoint to which the data is pushed.

Furthermore, several architectural requirements from a sustainability viewpoint are identified in a number of federations [70], [77]. The selected requirements relevant to monitoring are as follows:

- It is preferable to minimize the central components as much as possible as these would otherwise put the federation at a great risk concerning scalability and long-term operability.

- Support for a fast, easy way for infrastructures to join and leave the federation is required. This is valid for the monitoring solution as it's part of the federation framework.
- The solution should be as tool-agnostic as possible, thus being easily extensible, allowing integration of additional tools (compatibility with the legacy tools).
- A monitoring solution should be capable of ensuring federation compliance, e.g. reachability and link bandwidth between infrastructures.
- Data should be provided in an interoperable way through common interfaces.

3.1.4. User Communities

Monitoring interests and requirements vary among user communities. In a federation of infrastructures of the same nature, such as cloud infrastructures, users might have similar interests in monitoring services. However, in a heterogeneous federation, such as Fed4FIRE, various user communities are involved that have different interests. Such diversity should be taken into consideration while designing the monitoring solution used in these environments.

Furthermore, the solution should have the ability to deal with various configurations and resource deployments across the federation. This is due to the fact that multiple, diverse deployments from different users might simultaneously run on a single infrastructure or across multiple infrastructures.

In addition to these non-functional considerations, several functional user requirements are identified [17], [86], [94], [107], [146]–[148]:

- Monitoring services need to be offered and requested on an on-demand basis.
- Domain-specific measurement should be supported.
- Users in experimental facilities require monitoring information regarding the offered resources. This allows them to choose the best resources for their needs. Furthermore, it allows them to distinguish between errors caused by their applications and services from those related to the testbed infrastructures.
- Users should be able to create, view, update, and terminate monitoring configurations related to their applications and services in real time.
- An easy method is required for users to store their own measurements during and after the service runtime for post processing and analysis. The data should be clearly identified and stored related to their applications or services (e.g. using unique ID).

- Measurement data needs to be provided to users on-demand, either on a per-request basis, or on a regular basis or even on schedule to provide the information within a given time interval.
- Users should be able to share their stored data with others (individuals or groups) or even open to the public.

Further non-functional requirements that are of interest to users [86], [107] are the following:

- It is desirable to reduce or avoid user involvement in the setup and configuration of monitoring service deployments.
- Users demand both secure access to their datasets and results, and that the privacy of their data is guaranteed.
- Data should be collected from different sources and provided in a user-friendly manner via common and standard interfaces or visualization tools.

3.2. Requirements Analysis

A range of diverse monitoring and measurement capabilities and requirements have been discussed in Sec. 3.1. These are analyzed and the common and relevant functional and non-functional requirements are identified in this section. The identified requirements are grouped into three categories: general, federator and user related.

3.2.1. General Requirements

This section includes those requirements that are valid or required in a generic context, i.e. by the monitoring stakeholders. The identified general requirements are as follows:

Req-1 **Cross-layer monitoring** Monitoring information in multiple granularities is required from low-layer metrics monitoring (infrastructure level, e.g. CPU) up to the upper-layer (application specific) [26], [88], [94], [138]. This includes high-level information on facility status (e.g. resource availability for FLS), as well as monitoring of shared resources (e.g. from cloud infrastructures, either underlying resources shared by VMs or those shared by multiple applications within one VM [27]).

Req-2 **Cross-domain monitoring** Cross-domain monitoring is needed for a solution that is capable of operating across federated domains, gathering data from different sources and providing it to consumers in a common way [17], [88], [94], [138].

Req-3 **On-demand** Monitoring services are provided on an on-demand basis [64], [94], [143], [149].

Req-4 **Autonomic** A good monitoring solution should be able to automatically adjust to dynamic changes in monitored entities, as resources are created and destroyed in a dynamic manner. For instance, they should automatically react to the auto-scaling and rapid changes in the deployment of VMs without the need for any manual intervention [26], [39], [88], [123]. In federated infrastructures, there is a need to deal with changes in the federation, as MPs as well as monitoring tools will change and vary. The federation may be expanded to include more infrastructures or even shrunk [26], [123], [149], [150].

Req-5 **Comprehensiveness** A comprehensive solution is desired to support monitoring different types of resources and to collect several kinds of monitoring data from heterogeneous data sources (e.g. platforms, hardware, software, mobile or stationary devices) [26], [88].

Req-6 **Extensibility** An extensible solution is required to cover more features, as well as measurement metrics [88], [148], [150].

Req-7 **Scalability** The solution should be able to deal with a rapidly increased number of MPs and collected data [27], [31].

Req-8 **Standard and common data representation** Data has to be provided through standard and common formats and metrics [39], [88], [109]. A well-structured data model is needed in order to provide monitoring data in a meaningful way. It should cover the common concepts and relations of monitoring functionalities. It should also be extensible, so as to be able to accommodate new monitoring concepts of additional domains in the future [149].

Req-9 **Programming interfaces** Monitoring data needs to be provided in multiple ways, including visualization capabilities and APIs [94], [149]. It's desirable to allow users and tools to interact with monitoring capabilities through programmable interfaces [138].

Req-10 **Standardization and openness** Data needs to be transferred through standard interfaces and communication protocols federation-wide [39], [138], [149]. The adoption of the widely-used standards (e.g. Hyper Text Transfer Protocol (HTTP), SNMP) should also be possible.

Req-11 **Interoperability and compatibility** A domain-agnostic monitoring solution is desired that can operate across multiple infrastructures in an interoperable and compatible manner [123], [138].

Req-12 **Continuous monitoring** A continuous resource (software and hardware) monitoring process is desirable [138]. It supports fault management as it enables timely prediction and detection of failures (e.g. failure could happen in an overloaded server) that can be proactively handled [27]. It has also to be able to differentiate between failures and VM terminations [26].

Req-13 **SLA monitoring** Monitoring QoS parameters and further computation metrics is required for SLA validation [11], [138], [145].

Req-14 **Time sensitivity** Data should be provided with minimal latency, i.e. minimizing delay and transfer costs [26].

Req-15 **Accuracy** Data should be provided with high accuracy and precision [31], [123].

Req-16 **Reliability** In large-scale, distributed environments where huge amounts of data are exchanged, delay in data delivery and single point of failures should be avoided. A distributed approach might help in this matter.

Req-17 **Archivability** Historical monitoring data needs to be archived and thus remain available for post-processing [27].

Req-18 **Security** It's required to have secure data transmission, as well as authenticated access to monitoring data [27], [77], [150].

Req-19 **Minimal impact** As few overhead as possible should be expected from the monitoring solution [150].

3.2.2. Federation Requirements

Federation of multiple administrative domains is not a trivial task. It even introduces new challenges that need to be tackled as they arise. Further considerations have to be taken into account in order to cater for a sustainable and manageable federation. Therefor the federation adds more and new types of requirements to the previous ones. The identified federation-related requirements are as follows:

Req-20 **Tool-agnostic** The collection and representation of monitoring data at infrastructure level have to be independent from the monitoring tools installed [11], [77], [149].

Req-21 **Reusability** Monitoring tools that are already in place in the infrastructures need to be maintained. They can be enhanced or extended to adapt to the overall solution [77], [143], [149].

Req-22 **Federation aware** Monitoring solution components should be integratable into the federation architecture to interwork and interoperate with the rest of the components of the architecture, such as the management, control and federation systems [77], [149].

Req-23 **Real-time and historical data** Monitoring information about resources is required by some consumers (e.g. SLA management) during their usage (in real time), while other consumers (e.g. reservation brokerage) require historical information [149].

Req-24 **Data delivery** Monitoring data needs to be delivered in both push and pull manners [149], [150], as well as on a per-request basis and on a regular-basis [150].

3.2.3. User Requirements

As users are always at the heart of any infrastructures, further requirements from the user's viewpoint are considered [17], [86], [94], [107], [146]–[148]:

Req-25 **Ease of use** Automatic setup of monitoring services is required, allowing users to request only services of interest without involvement in the entire setup process. The setup and delivery of data are then the responsibility of the monitoring solution.

Req-26 **Usefulness** Users require the ability to setup and control multiple, diverse configurations that run simultaneously.

Req-27 **Customizability** A customizable solution is required that allows users to choose metrics they want to be monitored or even define new ones for monitoring their services or applications [27], [151].

Req-28 **User-friendliness** Access to monitoring data should be user-friendly.

Req-29 **Data storage flexibility** Users need to have the flexibility to decide where to store their data and how large the storage capacity should be.

Req-30 **Data availability** Users require access to data both during the services' lifetimes and afterwards for post-processing.

Req-31 **Group support** Support for the concept of groups is required, so that a group of users can manage their monitoring configurations and access the data as long as they are authorized.

3.3. Discussion and Gap Analysis

The rapid development of new ICT technologies and computing and networking paradigms requires suitable (maybe new) tools and methods for supporting their functionalities, such as monitoring and controlling capabilities. For instance, with the introduction of Cloud Computing, a suitable monitoring solution was needed that provides the necessary information for controlling and managing the deployed cloud services in accordance with standardized models and cloud characteristics (Sec. 2.2). Solutions used for monitoring grid infrastructures and distributed systems have been also used for monitoring clouds. However, multiple research studies on monitoring cloud environments [26], [31], [152] have shown via thorough analysis and comparisons of state-of-the-art solutions that none of them fulfills all requirements of cloud monitoring. Nevertheless, through enhancements or extensions, some tools provide most of the needed support.

The federation of multiple administrative domains is another active field of research in the ICT world, in particular in Cloud and FI experimentation domains (Sec. 2.4). Such a federation introduces further challenges and requirements to support its functionalities. The monitoring of heterogeneous resources across multiple administrative domains is recognized as one of most critical challenges for establishing a federation [21], [22]. The variety of the solutions used leads to several research questions related to compatibility and interoperability, as solutions could have specific design, various interfaces and use different protocols. The research literature lacks a solution capable of overcoming the limitations of state-of-the-art solutions in fulfilling the federation requirements. The need for such a solution has been identified in a couple of research projects that focus on designing federation architectures and, accordingly, building federated environments [11], [17], [28]; in research calls [29], [30]; and in the survey on cloud monitoring in [31].

As this thesis focuses mainly on facilitating the monitoring of federated infrastructures, the related requirements are considered. A list of monitoring requirements were identified in the previous section. Also considered were partial requirements from some emerging technologies that are at the heart of the concerned federations. That means that, on the one hand, those requirements that might be affected or need extra care in the federation are considered. On the other hand, other domain- or technology-specific monitoring related challenges and requirements might need further investigation, but these are out of the scope of this thesis.

Some of the identified requirements that support a particular functionality can already be supported by traditional tools. They are considered in the target solution delivered by this thesis as they are valid and significant in the federation. For instance, the monitoring solution has to be autonomic in order to deal with the rapid and dynamic changes in cloud resources, as cloud services might be offered in an auto-

scalling manner. This requirement needs to be further considered in a federated environment where resources are deployed across infrastructures. Other examples include archivability, scalability and security.

By contrast, some requirements are federation-specific and have not yet been considered by sate-of-the-art solutions in the types and scales of the federations addressed within this thesis. For instance, the monitoring solution should be tool-agnostic to facilitate building sustainable and dynamic federations. Further examples include interoperability and compatibility, and standard and common data representation.

The target solution should cover cross-layer and across-domain monitoring services provided for individual, diverse users on-demand. It should also reflect the general requirements that any monitoring system should consider, such as scalability, extensibility, and data delivery models (per-request or on-regular) to provide static hardware and software configurations (e.g. number of CPUs) and dynamic information (e.g. current CPU load) [150], in common data format and representation. To provide the data in a common format across heterogeneous infrastructures, a standardized protocol has to be used. However, meaningful common data representation is a painful task, in particular, where a large number of diverse metrics, tools and domains are considered. In order to have a common understanding and representation of concepts and relations in such a complex environment, common information and data models need to be used. Some of the models available (e.g. PerfSONAR, MOMENT and NOVI) are limited to particular domains and have narrow scope. In contrast, federated infrastructures, like those addressed within this thesis, have heterogeneous domains and wide scope. Such limitations of the previously existing models hinder their direct adoption; nevertheless, they are used as a starting point for the model developed within this thesis.

3.4. Summary

This section discussed various sets of monitoring requirements. Sec. 3.1 presented four types of requirements sources: i) emerging, thesis-relevant technologies and networking paradigms, ii) testbed providers, iii) federation operators, and iv) users. These requirements are analyzed in Sec. 3.2 and a set of concrete functional and non-functional requirements are identified. The need for the selected requirements and their realization are discussed with a gap analysis in Sec. 3.3. All the identified requirements are considered in the design of the target monitoring solution, as will be discussed in Chapter 4 (architecture design and specification) and Chapter 5 (monitoring information model).

4 Architecture Design and Specification

THIS chapter introduces the design and specification of the monitoring architecture delivered by this thesis. Design was performed as an iterative and incremental activity. Based on the requirements identified in Chapter 3, it focused first on fulfilling the basic requirements for monitoring federated cloud infrastructures, considering various aspects of clouds (e.g. virtualization, elasticity) and the federation (e.g. different administrative domains with different access policies). The design has evolved according to additional requirements on flexibility and sustainability, as well as feedback from deployment of the initial design. Going beyond the requirements and use of the solution within the context of the cloud domain, a generic, flexible and extensible design has been targeted. This design covers various aspects for monitoring federated, heterogenous infrastructures that comprise domains of different natures (e.g. cloud, WiFi, cellular and sensor networks, SDN-enabled), taking the federation of heterogeneous testbeds for FI experimentation as a use-case.

After performing a thorough analysis of monitoring requirements (Chapter 3) and state-of-the-art solutions (Chapter 2), the first step towards achieving the objectives of this thesis is to design an architecture for monitoring federated infrastructures. The architecture should accommodate various types of monitoring and measurement services and provide the data for multiple stakeholders. The aim of this architecture is to meet the identified requirements of stakeholders, as well as support the specified, relevant functional and non-functional requirements, while taking into consideration infrastructures of different owners and natures.

Note, however, that this thesis does not intend to design and develop a completely new architecture from scratch, thus avoiding the introduction of yet another solution. Instead, the relevant existing technologies and protocols are analyzed and, as much as possible, adopted or adapted to be part of the architecture's functional elements. The missing architectural elements are specified and implemented. The architecture aims to fulfill all the demanded functionalities, taking into account multiple non-functional requirements, such as standardization, reduction of implementation effort and time, and performance.

The design process has been developed iteratively and incrementally, adapting to changing requirements (due to change of federation models and the federated infrastructures addressed in this thesis) and responding to feedback from the deployment of the initial design (see Figure 4.1). From this perspective, this chapter first discusses the initial phase of the design process (including methods and motivations behind decisions made during these phases), followed by the final architecture design and specification.

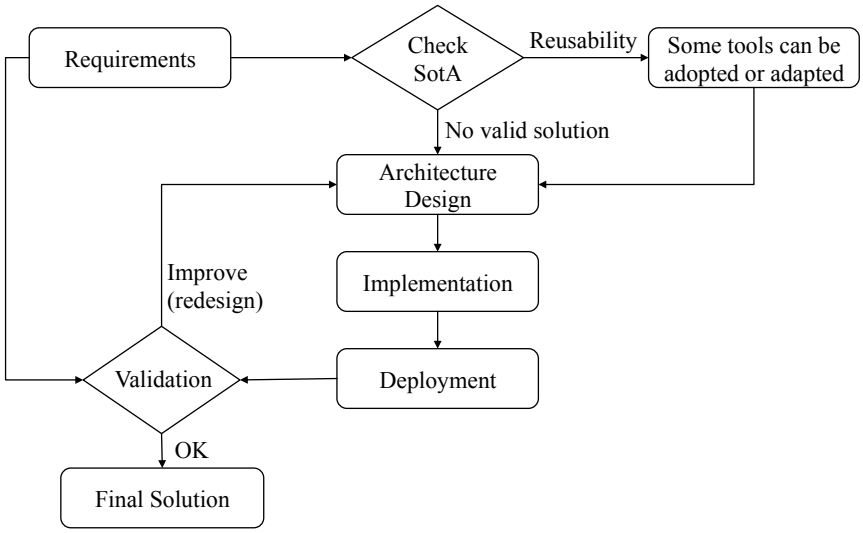


Fig. 4.1.: Design process

4.1. Conceptual Phase of Design

This section discusses the engineering design process in its initial phases. It's worth mentioning at this stage that the initial motivation for this thesis was the limitations of the state-of-the-art monitoring systems and the urgent demand for a system capable of aggregating a multitude of measurements from resources of differently administrated, federated cloud infrastructures [30]. This is due to the fact that each application, platform or infrastructure has its own monitoring solution and this hinders interoperability across federated clouds in terms of management, control and information exchange.

The design process is divided into two phases:

- **Initial Design Phase** The focus was on designing a system architecture that fufills a set of basic monitoring requirements for emerging cloud platforms, with an emphasis on their federation and the deployment of services and applications across domains in the form of PaaS and IaaS.
- **Final Design Phase** The focus was on improving the design towards a generic and flexible architecture taking into account sustainability considerations. In this phase, the solution assumes that each infrastructure participating the federation runs its native (or any suitable) software, but provides the data in a common way.

4.1.1. Initial Phase of Design

The design follows a homogenous federation model (see Sec. 2.4.1) and is based on the fact that the same monitoring tools are deployed on all infrastructures participating in the federation in order to facilitate interoperability and compatibility (Req-11). Each infrastructure runs an instance of the monitoring solution and is, thus, independent of any central components or organizations.

The architecture design is driven by a twofold objective. First, monitoring information for infrastructure providers and federation administrators should be provided to assure the health and performance of the federated cloud infrastructures and their internal services (e.g. supporting elastic service deployment), as well as service related measures that are consumed by cloud users to monitor their deployed cloud services. Second, the required services should be supported with minimal effort and complexity.

The term 'user' is used in this chapter and the following chapters to refer to a service provider in any ICT infrastructure, cloud customer in a commercial cloud or an experimenter in a cloud-based or FI testbed. Furthermore, a User-Customized Resource Environment (UCRE) refers to an environment that comprises a collection of resources (compute, network, storage, software, etc.) created in a customizable, on-demand manner, configured and used by a user or even a group of users. A UCRE

is used to run an ICT service in any ICT infrastructure or an experiment in an experimentation facility. In cloud infrastructures, a UCRE can be a cloud service (e.g. PaaS or IaaS). It can be also any ICT service platform that can be created by a user in a customizable, on-demand manner similar to the cloud service models or experiments in experimentation facilities.

4.1.1.1. Architectural Design Principles

In this design, it's assumed that the networking (on-site and cross-site connectivity) is managed at the federation level by using a Wide Area Network (WAN) or a layer-3 multi-domain Virtual Private Network (VPN). The architecture is then designed following a number of architectural principles [64] as follows.

Monitoring-as-a-Service (MaaS) Monitoring services are offered like any other cloud delivery models, following the concept of *XaaS*, and thus under *Monitoring-as-a-Service*. The architecture is capable of providing monitoring services on-demand. It leverages the paradigm of *everything-as-a-resource*. I.e. if it's a resource, it can be instrumented (through specific resource adapters, also called probes) and, as such, monitored (Req-5). Monitoring data provided by the probes is collected in the data collector and is finally accessed by users or other systems.

On-Demand Basis Users are able to request any of the offered monitoring services. This decision can be taken while requesting cloud resources of the UCRE (Req-3).

MaaS Provisioning During the deployment of a UCRE, the monitoring probes are deployed with the rest of the cloud resources as just another deployable object (Req-25, Req-26, Req-28). Before that, the deployment of the data collector is performed, which collects and stores data. All probes are then automatically configured to report data to the collector (Req-25, Req-26, Req-28). The measured data can then be accessed via its interfaces. The system design assumes that each user has a single monitoring collector running on one of the VMs created as part of the UCRE. This both assures privacy of information (Req-18), and is guaranteed to work with a wide range of cloud providers, as it provides monitoring as an over the top service in a cross-domain manner (Req-2).

Data Archive (Short- and Long-Term Availability) Users are able to have data permanently available after the expiration or deletion of their UCREs or only available during their lifetimes (Req-30). Three relevant cases can be distinguished:

1. Data is used to observe the correct progress of services or applications deployed in UCREs. There is no interest in keeping this information.

2. A user decides to keep a trace of the UCRE after its lifetime. As infinite disk space is not available, keeping all data indefinitely is not possible. Nonetheless, the full data for the last day (or whatever delay is compatible with the cloud providers' infrastructure) is available, for a limited time.

3. A user knows that the complete monitoring data will be useful. In this case, while establishing the UCRE, the user can ask for persistent storage with large enough storage size as needed (Req-17).

Data Storage Flexibility Users have two options as to where to store monitoring data (Req-29). It can be stored either inside the same resource hosting the collection server VM or on external storage resources. With the second option, the database of the collector is stored in external (and permanent, if required) storage that is mounted as an additional block device on the collector VM. This option enables more flexibility, a user can on-demand identify the required storage size and the data is available after the expiration or deletion of the UCRE.

Data Access Programmability Users have the ability to access their data through standard APIs (or at least in standard formats, such as JavaScript Object Notation (JSON) or Comma Separated Values (CSV)) as well as able to display the data graphically through a GUI (Req-8, Req-9, Req-10, Req-28).

Groups and Access Permissions A user has full permissions for their UCRE's monitoring collector. However, a group of users could work together on one UCRE and have the same permissions, but use different credentials. In this case, the monitoring system is responsible for enabling all group members to log in to the monitoring data using their own credentials (Req-31). Furthermore, limited access to monitoring data with read-only permissions can be granted to other groups in the case of cooperative projects (Req-18).

On-the-Fly Reconfiguration On-the-fly adjustments should be possible to accommodate the rapid and dynamic changes during the lifetime of the UCRE, as resources are removed or added. This is done through reconfigurable logic based on observing notifications and events and takes suitable actions to adjust monitoring services configuration and data delivery (Req-4, Req-7).

Extensibility The architecture should be extensible to include additional measurement metrics or even monitor new entities (Req-6).

Customizability Users should be able to define their own metrics to be measured according to their needs and specific to their services or applications (Req-27).

4.1.1.2. Cross-Layer Monitoring Services

These architectural principles enable the monitoring system to provide flexible monitoring services. They support cross-layer monitoring (Req-1), as shown in Figure 4.2 and described as follows:

Infrastructure Resources Monitoring Physical infrastructure resources of the federated clouds are monitored, considering various metrics that are used for assuring the health and performance of the infrastructure, e.g. CPU-, memory-, disk-usage, number of VMs running on each physical host, and ingoing and outgoing traffic.

UCRE Monitoring Metrics are monitored that address the UCRE status at a certain moment in time and the number of VMs per UCRE. This includes VM monitoring, i.e. their status and further metrics like CPU-, memory-, disk-usage, I/O operations, etc.

Services and Applications Monitoring Metrics are monitored that provide information about the state of the service or application, its performance and other service or application specific information.

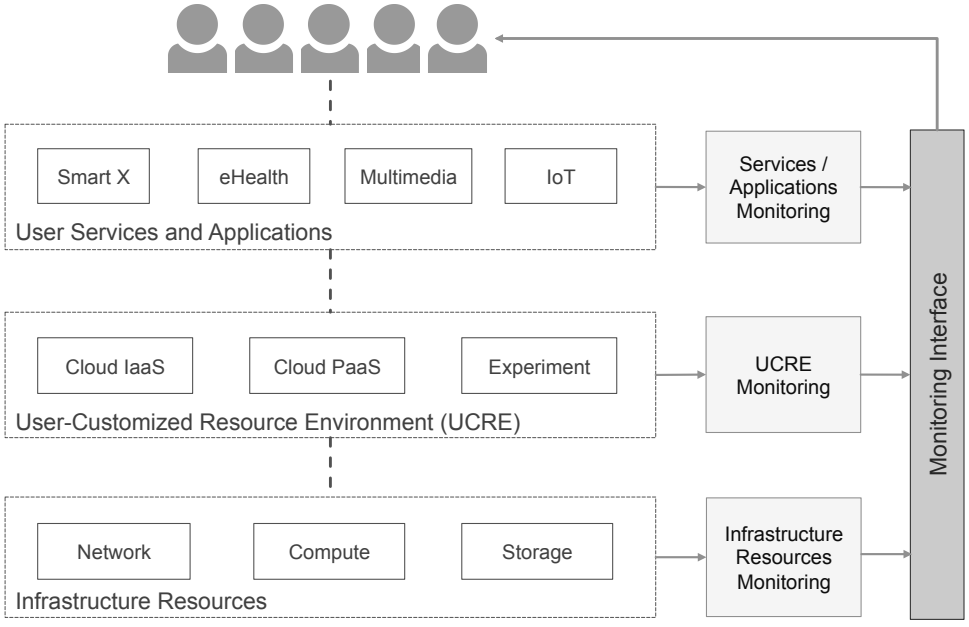


Fig. 4.2.: Cross-layer monitoring services

Concerning the last two monitoring types, namely UCRE monitoring and services and applications monitoring, users have full control of the monitoring software and setup and can configure it as described. They can monitor some or all used VMs, the network performance between them, QoS parameters, etc. They can monitor

the performance and behavior of the deployed services and applications through the *customizable capability*.

In addition, a user can get partial information about the physical machines that host their VMs, referred to as infrastructure monitoring. Monitoring infrastructure resources is the responsibility of the infrastructure providers. However, to provide this service for users and federation administrators (and probably for internal use), an infrastructure monitoring collector is required that continuously collects information about the whole infrastructure as reported by the probes running in all physical machines (Req-12). Information about specific metrics is only provided for users, according to of users' interests, such as CPU utilization, memory consumption, number of running VMs and network characteristics.

4.1.1.3. Initial Architecture

The monitoring architecture is shown in Figure 4.3. It is designed based on a client-server model. Monitoring probes, also called clients, are deployed and configured on each of the user's VM resources. These clients are responsible for collecting data for measurement metrics on their respective hosts. Information is sent to a user monitoring collector (User Collector), which holds all monitoring information, for direct consumption by the user (via an API or a GUI). The user collector representing the server is deployed as a separate resource (part of the UCRE) and collects monitoring data reported by the clients in the UCRE's VM images.

In addition, the user can get monitoring data about the physical machines hosting VMs from the respective infrastructure's monitoring collectors (Infrastructure Collectors) through their APIs. This is achieved through a daemon running on the user collector that is responsible for periodically fetching data and their timestamps and then storing the data in the database of the user collector. When user VMs are deployed in multiple clouds, their infrastructure collectors are then accessed by the user daemon. This daemon is notified each time a VM is created, updated, or destroyed. The notification is delivered through the relevant model, such as message queuing and publish-subscribe interaction. The use of message queue allows guaranteed notification delivery. The daemon acts as a listener and subscribes to notifications sent when VMs' states are changed. Once a VM is created, the daemon receives notification, parses the content and stores this information in a local database along with the name of the physical machine hosting the created VM and the cloud infrastructure where it's deployed. The daemon then fetches infrastructure monitoring information about this physical machine from the respective infrastructure monitoring collector. If a running VM is destroyed, it's removed from the local database and no monitoring data about its hosting physical machine is fetched, unless the user has other VMs hosted on the same machine. The user can then get data through the API or the GUI of the user collector with the original timestamps.

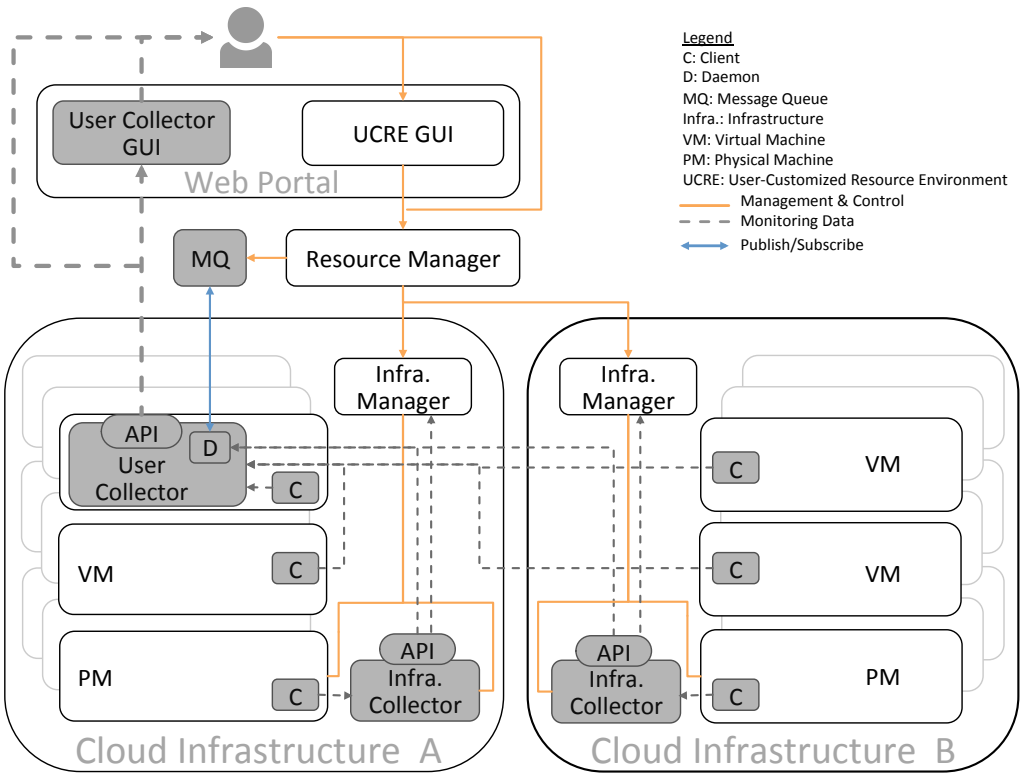


Fig. 4.3.: Initial design of the monitoring architecture (based on [64])

The technical details of gathering metrics pose a scientific challenge themselves. Moreover, it is virtually impossible to predict all conceivable monitoring metrics that might be of interest for cloud users, especially when addressing scientific users. Therefore, any comprehensive cloud monitoring solution must be extensible through user-defined metrics. For this reason, the architecture is extensible in the sense that it allows users to define their own probes or metrics on the client sides in order to observe any software or hardware and get the desired information.

4.1.1.4. Architectural Limitations Based on Experience

This architecture was adopted and implemented for monitoring a large-scale, multi-site, federated cloud facility for FI services experimentation within the European project BonFIRE. BonFIRE consists of six cloud infrastructures geographically distributed across Europe offering heterogeneous cloud services for users, who develop and trial FI-oriented services and applications.

Although this architecture supports cross-layer monitoring and aggregating a multitude of measurements from resources of different administrative clouds in a unified manner (details in Chapter 7), it assumes the same monitoring tools are deployed in each of the federated infrastructures.

This architectural limitation prevents straightforward integration of cloud infrastructures with their own monitoring systems in place, according to a study performed and published in [153]. In this study, the integration of an OpenStack[11] based cloud infrastructure within the BonFIRE federation was investigated. This infrastructure uses the native OpenStack tools and APIs for observing events, messaging and notifications, as well as controlling the infrastructure resources. Issues like event observability, controllability and API compatibility were considered. The study showed a possible integration through translators and adaptation components. However, this is always required for any further infrastructure integration. Besides the effort spent for implementing such mapping, mapping procedures have to be defined and realized anew each time due to the absence of standardization means.

From this perspective, an open, tool-independent architecture design is essential (Req-20). This means that the monitoring solution should allow each federated infrastructure to keep using the tools already in place or use any suitable tool for its practical needs (Req-21). However, the diversity of the tools used raises a heterogeneity problem concerning the data formats and schemas, and the ways the data is represented or accessed. It is therefore necessary to define a mechanism to harmonize data collected from different sources and then provide it in a unified manner.

4.1.2. Design Decisions and Goals for Final Design

The major architectural changes in the final design compared to the initial one are that it assumes the use of heterogeneous monitoring tools at infrastructure level and adopts a proper mechanism responsible for enabling a uniform representation of monitoring data across domains in order to overcome the heterogeneity problem (Req-8 and Req-11).

Solving the heterogeneity problem is not a trivial task. Nevertheless, after extensive analysis of possible solutions, a proper design is deemed to follow a federation model, where each infrastructure adopts a specific set of common APIs and adaptation mechanisms to become part of the federation and offers compatible services independent of the infrastructure to support the entire service (or experiment) lifecycle including resource description, discovery, provisioning, monitoring and control. Pros and cons of this approach and other evaluated ones are presented in [10], and discussed in Sec. 2.4.1. To this end, it is necessary to define those APIs and mechanisms that allow an infrastructure to offer the full capacity of the federation services in a unified manner, and to achieve the necessary degree of compatibility with the rest of participating infrastructures (Req-22). As this thesis focuses on monitoring aspects, the related interface and adaptation mechanisms are thus described in the following section.

From this perspective, such design allows any infrastructure to be easily integrated as long as it adapts its tools to the common interfaces in order to provide the data in

a unified manner. Furthermore, if a tool is already adapted to a common interface, it can be deployed in other infrastructures that have no tools in place. This allows different infrastructures to benefit from the development efforts of others.

The advantages of such a design are multifold:

- Infrastructures don't need to replace their monitoring solutions but can keep their own and adapt to the common interface.
- Reduced time and effort is needed to be part of the federation.
- All stakeholders in the federation speak the same language through the common interface.
- Infrastructures can continue operating and serving their legacy users, as well as those from different federations (in case they are part of multiple federations) through different interfaces.
- An infrastructure can smoothly leave the federation and continue operating as it was before being federated, since it keeps using the same software.

4.2. Generic, Flexible and Extensible Architectural Design

One of the main architectural principles for designing the monitoring architecture is to define an adaptation mechanism (as an intermediate layer as shown in Figure 4.4) for mapping from heterogeneous monitoring systems to the *Common Monitoring API*. This is achieved through a unified reference model that shall be implemented by the different infrastructure managers specific to their monitoring tools in terms of *adapters* (also called *wrappers*). Adapters are responsible for managing monitoring tools, i.e. interacting with local monitoring tools. Functionalities of an adapter can include processing and aggregation of raw measurement data, and generation of new data following common information and data models. For each monitoring service offered by the infrastructure, at least one adapter is needed, i.e. various adapters should be implemented. The functionality of each adapter differs depending on the type of monitoring service it is responsible for. Therefore, an adapter can be in charge of all or some of the following processes: installing software, running software, processing, collecting data, and pushing data in a specific format to a data collector.

Within this thesis a set of adapters is defined that is in charge of providing an abstraction away from the diverse set of monitoring tools deployed by the infrastructures participating in the federation. This abstraction is referred to as an adaptation layer. The abstraction layer aims to collect monitoring data from the federated infrastructures, and publish it in a unified and standardized manner through the common

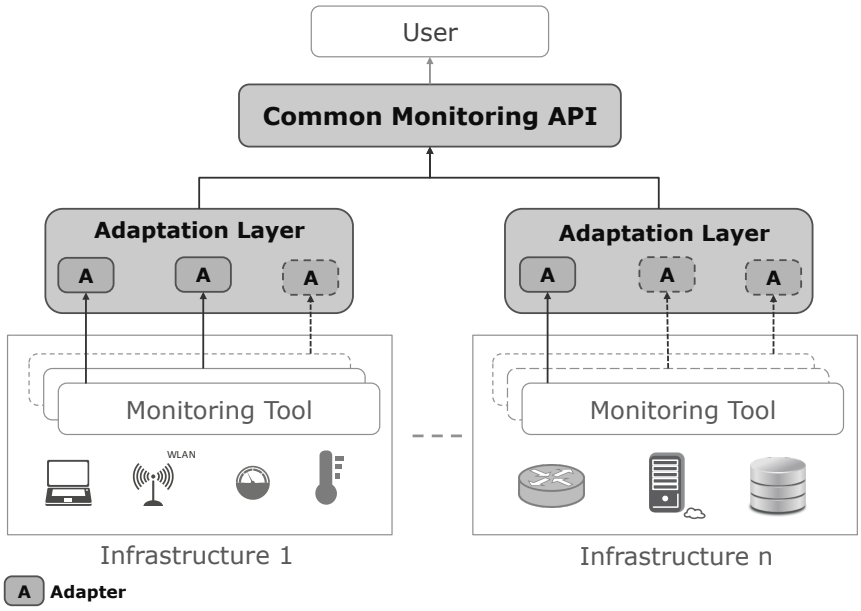


Fig. 4.4.: High-level representation of the common monitoring API

monitoring API (Req-8). For this to be realized in a meaningful and standardized way, an ontology-based information model was developed within this thesis. It provides formal, semantic descriptions of the common monitoring and measurement concepts and their relationships in federated infrastructures focusing on the areas of interest in this thesis.

The solution architecture is designed to be extensible and capable of accommodating further monitoring tools through additional adapters, as depicted in Figure 4.4 through the dotted boxes and lines.

4.2.1. Architectural Design Principles

Besides the architectural principles discussed in Sec. 4.1.1.1, the following principles were fundamental throughout the final design process.

Cross-Domain Compatibility Cross-domain compatibility (Req-11) should be supported so as to act as if the data is being provided by a single domain. This requires a suitable mechanism capable of collecting and representing the data across multi-domain infrastructures in a unified manner.

Open APIs Data should be provided through a common, open API (Req-10). Open interfaces (for resource provisioning, monitoring, and control) in a federation environment enable different stakeholders (infrastructure providers, federators, and users) to be part of an interoperable ecosystem and mobilize others (service providers

and application developers) to join, thus facilitating the establishment of a wider community.

Data Modeling A well-structured data model should be defined to deliver data in a meaningful way (Req-8).

Data Delivery Data delivery needs to be in both the following modes (Req-24):

- on a per-request basis, i.e. on user request, and
- on a regular basis, i.e. data is provided periodically, including even real-time delivery.

Interoperability Monitoring components must be compatible and interoperable with the remaining components of the federation architecture, such as the provisioning and control components and the federation services' systems (Req-11, Req-22) .

Scalability Capability to deal with the rapid and dynamic changes in the federation is required due to the increased number of MPs and diversity of tools used (Req-7).

Reusability and Extensibility The architecture is designed in a way to allow reuse of existing adapters that have been developed as long as they serve the same subject matter (Req-21). It is also easily extensible to accommodate additional monitoring and measurement metrics (Req-6).

4.2.2. Reference Federation Model

After evaluating different federation approaches [10], as discussed in Sec. 2.4.1, the architecture design follows a heterogeneous federation approach, where infrastructure providers, developers and users interact through common, open APIs. This approach has been adopted in several large-scale federations built by Europe's largest programs within the context of FI experimentation [11], [70]. However, the reference federation followed in this thesis is shown in the very abstract and high-level overview in Figure 4.5, also reflected in Figure 1.5.

Figure 4.5 includes three infrastructures offering different types of resources and using different tools for their management. However, they offer and interact with the outside world in a common way (through common APIs supporting resource description, discovery and provisioning, monitoring and control). Users on the other side use any tools or software able to speak the common languages. For simplifying the model, this figure does not show all federation aspects but only those of major concern within this thesis. For instance, identity management is one of the key aspects in any federation and is required at all stages of a service (or an experiment) lifecycle.

Furthermore, federation administrators should have high-level monitoring information about the health and status of the federated infrastructures in order to take proper actions early on.

Such a federation model enables any stakeholder to join or leave the ecosystem easily as long as it adopts the common interfaces, which should be open and well-defined. Furthermore, the federation can be realized in a distributed manner, where users can interact directly with the infrastructures as long as they are authenticated and authorized to do so.

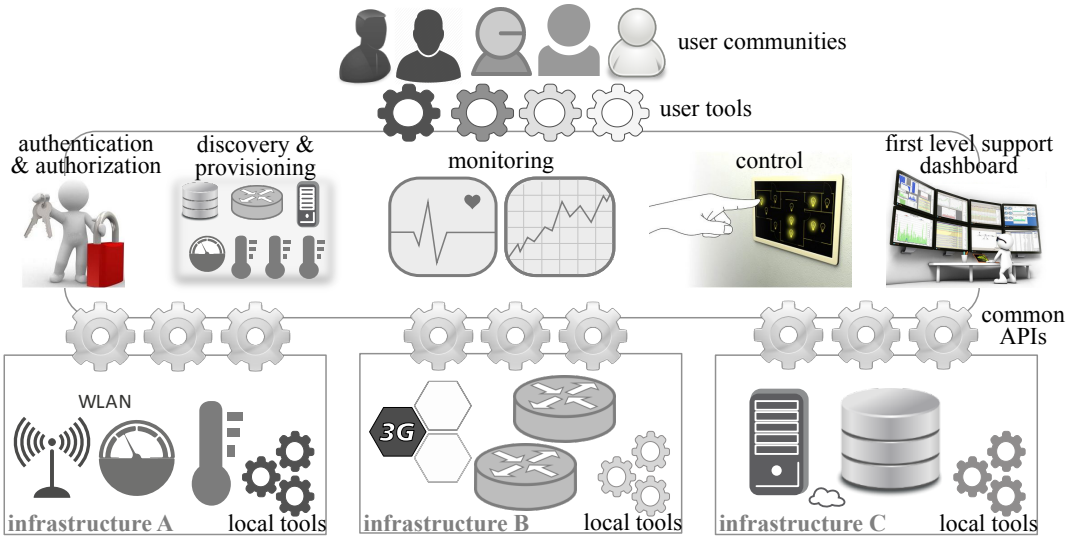


Fig. 4.5.: High-level overview of the federation reference model followed in this thesis

4.3. MAFIA: Monitoring Architecture for Federated Heterogeneous Infrastructures

This section presents and discusses in detail the final version of Monitoring Architecture for Federated heterogeneous Infrastructures (MAFIA).

MAFIA is described to be **generic** so as to handle any kind of measurements produced through any kind of monitoring system within any domain type. Furthermore, it can be used to operate across any kind of ICT infrastructure federation (cloud-based, fixed and mobile broadband telecommunication networks, or heterogeneous domain federation). It can be therefore described as **adaptable**. MAFIA is referred to as **flexible** due to a design that enables support for numerous requirements with different possible implementations depending on the type of the offered monitoring service and its consumption. It is flexible in the sense that it can be deployed with minimal effort required from the infrastructure owners to be compliant to its requirements. This is because they can keep using their local monitoring systems to provide the concerned

services as long as they provide the data following the common API. Moreover, it is not mandatory to provide the same set of measurements across the federated infrastructures but each can decide which measurements are of major concern for their users. MAFIA is **comprehensive** due to its ability to deal with large amounts of data from multi-domain and across-layer monitoring solutions provided through a common API for multiple stakeholders. Finally, MAFIA is **extensible** to accommodate additional MPs or monitoring further ICT domains in the form of adapters as illustrated in Figure 4.4.

To simplify the design and implementation of the architecture, monitoring and measurements services in federated infrastructures (focusing on Cloud and FI experimentation areas) are classified [143].

4.3.1. Types of Monitoring and Measurements Services

The initial design introduced three types of monitoring services (see Sec. 4.1.1.2) that were defined according to the requirements at the initial stage of work in this thesis. However, in later stages, and according to the requirements analyzed in Chapter 3, the final design has considered and classified monitoring and measurements services into four different categories that differ from one another according to level and granularity (Req-2). These types, discussed in the following sections are:

- Infrastructure Health and Status Monitoring;
- Infrastructure Resources Monitoring;
- UCRE Monitoring; and
- Services and Applications Monitoring.

Figure 1.5 illustrates these services, which together represent cross-layer monitoring. Note that Figure 1.5 shows only those services deemed to be of major importance to the areas of application addressed in this thesis, namely Cloud and FI experimentation. However, further monitoring services, classifications, and even monitoring new areas or domains are possible as the architecture is designed to be extensible. This is discussed later in the implementation and evaluation chapters 6 and 7.

4.3.1.1. Infrastructure Health and Status Monitoring

This type of monitoring provides twofold information about an infrastructure facility, namely, low-level information about its individual resources and services for internal operational and administrative need; and high-level information about the availability of the facility, which is exposed to the outside world.

Low Level Infrastructure Monitoring This service represents monitoring of the resources and networking services of the whole infrastructure. It is required by the infrastructure provider in order to ensure the health and performance of its facility, as well as to ensure the availability of all its offerings.

High Level Infrastructure Monitoring This service provides high-level information about the availability of infrastructure facilities, namely whether each is up and running, in risk, or down. It allows federation and system administrators or FLS operators to verify that facilities are performing correctly, and verify their connectivity and interworking. RAG status is pushed to a central dashboard for FLS reactive monitoring. The infrastructure RAG status is based on the monitoring of key components of each infrastructure that indicate the overall availability status of the infrastructure facility. However, through the availability status information provided, FLS operators are able to drill down from the infrastructure RAG status to see which components are degraded or down. Status information depends on specific monitoring metrics of the key components of the infrastructure facility. These vary from facility to facility depending on the nature of each infrastructure in terms of the types of services and resources offered.

4.3.1.2. Infrastructure Resources Monitoring

This service provides information about infrastructure resources for users and federation services. Types of monitoring information and metrics vary from infrastructure to infrastructure based on the nature of each infrastructure and its offerings. A practical example of such is seen in the heterogeneous infrastructures involved in Fed4FIRE federation. Under this monitoring category, two types are to be distinguished: resource characteristics and information per service basis.

Resource Characteristics Quality of the resources offered may vary from one to another and also from one infrastructure to another. To increase user satisfaction by providing information about offerings, measurable characteristics (real time or historical that do not correspond to a certain UCRE) of resources might be useful for resource selection (Req-23). For instance, a user might require a set of nodes that have good quality pairwise connections. In a wireless network, these might be nodes that have good measured signal strength between each other. In a distributed server infrastructure, these might be nodes that have Internet routes between them that have been determined to be stable over time. Furthermore, such information is useful for the reservation brokerage service.

Information per Service Basis This type differs from the previous one in the time interval and type of data delivery. It delivers information on a per-service basis

about infrastructure resources (networks, infrastructure services and nodes) to which users have no access or cannot instrument themselves but which are relevant to their UCRE resources (Req-23). Per-service basis means that this monitoring service delivers information only about the infrastructure resources used by the UCRE during its lifetime. It is required by the users and some federation services.

- ***For Users*** Monitoring environmental conditions in the service development lifecycle and in experimental driven research is of major importance. Looking around the deployed service (or experiment) setup for influencing factors in the infrastructure (e.g. virtualization that may increase delays due to scheduling, bad behavior of co-existing VMs used by other users) might increase the evaluation accuracy of the service performance and experiment results. Infrastructure resource information of users' interests could be, for example, about physical host performance if the user uses VMs, or wireless spectrum, among others.

- ***For Federation Management and Operation*** Federation services might need information about infrastructure resources for their specific purposes. For instance, information about specific predefined measurement metrics is required by SLA management to validate SLAs (Req-13). A trustworthy reputation service requires information about particular resources and services that were used during the UCRE lifetime (on a per-experiment basis) from both infrastructure providers and the user in order to calculate reputation scores and finally provide high level descriptions of the quality of the offered resources and services. This knowledge is useful for resource selection by users.

4.3.1.3. User Customized Resource Environment Monitoring

This type of monitoring service represents monitoring UCRE resources that are used to run a service or an experiment and are under the full control of the user. This includes looking within the UCRE setup, hardware and networks, operating systems, and component related metrics, such as CPU, memory, disk utilization, and bandwidth.

4.3.1.4. Services and Applications Monitoring

This type of monitoring service represents measurement metrics that provide information about the state of the service or application being tested, its performance and other specific information. This includes custom metrics defined by the user (Req-27) for observing the service or application in progress and for its evaluation (e.g. to show improvements and comparison to other approaches). Custom metrics are heterogeneous and vary depending on services or applications being tested. Examples on a per service/application basis include CPU and memory consumption, errors and warnings, and end-to-end service quality.

4.3.2. Architecture Components and Interactions

This section describes the architecture components and their interactions, as well as the various monitoring services supported by MAFIA.

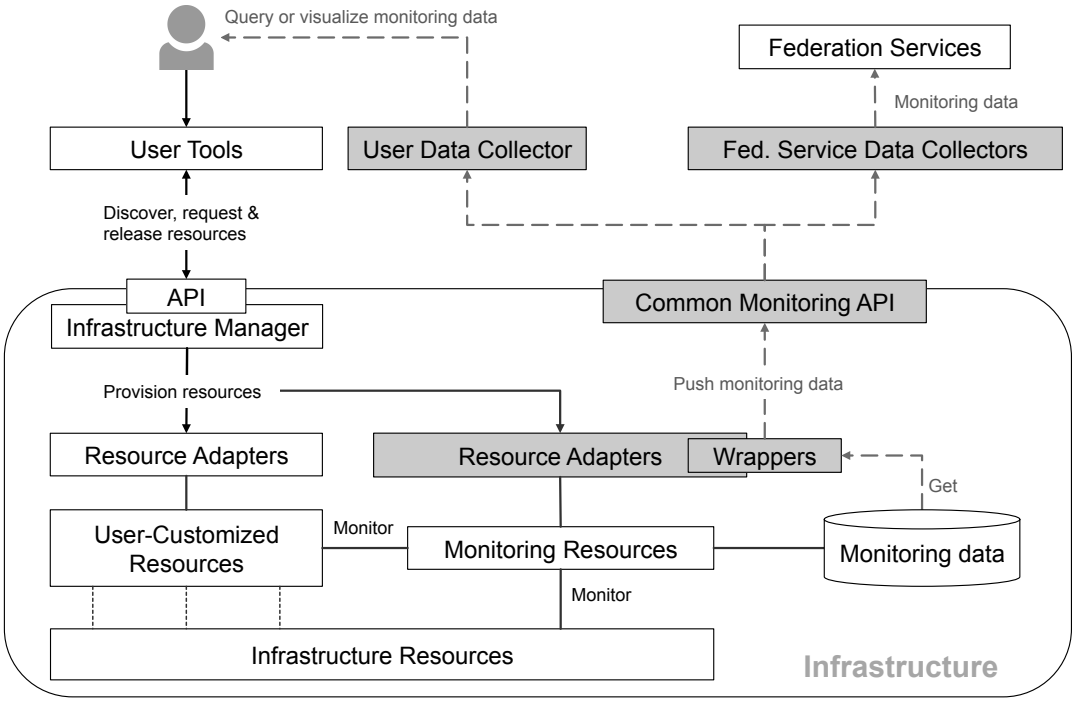


Fig. 4.6.: MAFIA components and interactions (based on [143])

Figure 4.6 shows an overview of MAFIA components and their interaction. The components in charge of data unification and collection are highlighted in gray. In order to simplify the figure, only one user with one data collection resource (user data collector) is considered. Similarly, only one infrastructure is illustrated. Nevertheless, the same deployment is to be expected in the remaining infrastructures participating the federation. The data provided by all infrastructures is collected in the user data collection resource. Collectors for data provided for federation services might be deployed locally at the infrastructure level or centrally at the federation level depending on the deployment of each service. If a federation service is deployed in a distributed mode, which means an instance of its logic is deployed in each infrastructure, it is recommended to have the collector at the infrastructure level as well. This will be discussed again in detail in Sec. 4.3.2.3 and in Chapter 6.

The functionalities and roles of MAFIA components are described in the following sections through the description of MAFIA support for the concerned monitoring services discussed in Sec. 4.3.1.

4.3.2.1. Monitoring Services for Users

Users are interested in information about infrastructure resources at two different levels of granularity.

First, users look for performance and quality characteristics about the offered resources in advance before requesting resources. Such information is provided with the resource advertisements. Advertisements for infrastructure offerings are made available to the federation community through a common API exposed by the *infrastructure manager*. This API is used by users to discover, request and release resources required to setup their UCREs.

Second, users are interested in information about the infrastructure resources used by (or related to) the resources deployed to run their services or experiments. This is due to the fact that understanding the surrounding conditions helps for a more accurate evaluation of the results.

Methodology for how the second case is supported, together with UCRE monitoring, and services and applications monitoring, is discussed in the following sections in two different possible implementation approaches, namely *generic* and *user-friendly*.

Generic Approach:

Monitoring information is collected from monitored resources across infrastructure facilities, stored in data collection resources, and is then accessed by users. Usually monitoring goes through multiple stages: data acquisition, collection, transportation, storage, access, and archiving. Accordingly, various monitoring resources are used: agents and probes (e.g. Zabbix agent, Nagios agent, Iperf, spectrum analyzer) for data acquisition; data collection and transportation resources (e.g. SNMP, OMSP, IPFIX); data collector (e.g. SQL server, OML server, Jena Apache Fuseki[1]); storage resource (e.g. MySQL, PostgreSQL, SQLite, Triple Store); archival resource (permanent storage resource); and data broker or access resource (any data retrieval tool).

The generic concept of MAFIA architecture is that infrastructures offer such monitoring resources as normal resources similar to the main offered resources (e.g. VMs, sensors, Wi-Fi nodes, platforms) [94], [143]. Due to the heterogeneity of the offered resources, infrastructures might offer different types and numbers of monitoring resources specific to their domain offerings. From this perspective, users have to be aware of the functionality of each resource, and accordingly request and configure the required resources and their relationships.

Therefore, users can setup UCREs together with monitoring services provided by individual infrastructures. This covers the monitoring of infrastructure resources, UCRE and user services and applications. This is done after user authentication as illustrated in Figure 4.7, which shows a sequence diagram for the whole procedure.

[1] http://jena.apache.org/documentation/serving_data/

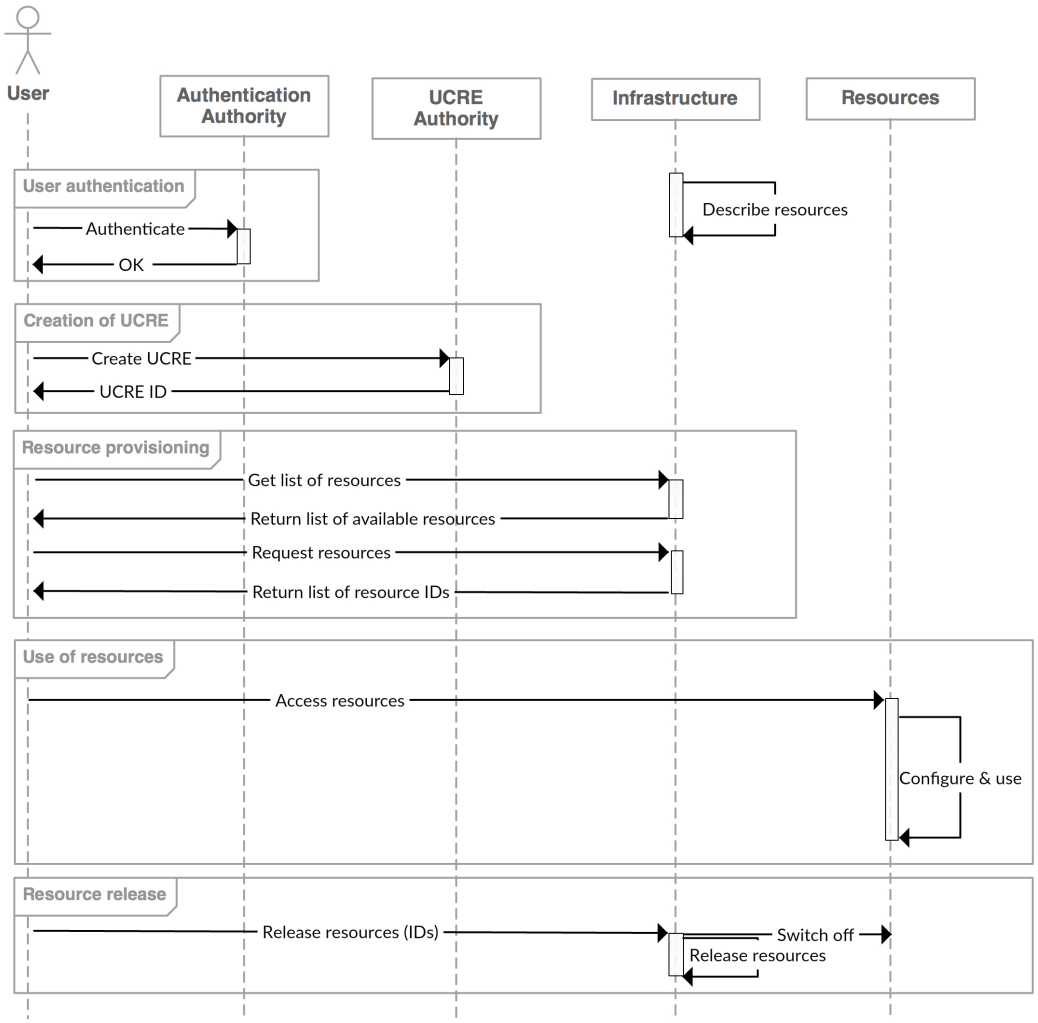


Fig. 4.7.: Sequence diagram for the setup of monitoring services in the generic approach

Using any user tool, a user can discover and request monitoring resources along with the required resources for setting up a UCRE across infrastructures. Individual requests are dispatched to the respective infrastructure managers that are in charge of resource provisioning. For each resource, a resource adapter is used that is in charge of its initiation and configuration. For monitoring resources, the roles of each resource adapter differ from one to another depending on the type of monitoring resource. Among its roles, a resource adapter can be responsible for all or some of the following functionalities: software installation, configuration, running, processing, performing measurements, data collection, data conversion and data expiration. The data is provided to the user in a unified data representation via the common monitoring API. This is achieved by wrapper resources. Wrappers are in charge of retrieving data from local tools and converting the data from local formats and publishing it in the common format supported by the common API. Wrapping functionality can

be one of the functionalities of a resource adapters, therefore Figure 4.6 shows an overlap between wrappers and resource adapters. Data is finally collected in the user collection resource. The user collection resource can be owned by the user or offered by any of the federated infrastructures and can be provisioned like other resources.

This high level of abstraction of the architecture simplifies implementation efforts from the infrastructure provider's prospective, who would otherwise need to setup the whole monitoring processes for the individual UCREs. On the other hand, the user might need to know the functionality of each monitoring resource and then request those of interest. Furthermore, users themselves are responsible for setting up the required monitoring services, i.e. by requesting a set of monitoring resources that together realize a monitoring service. Such service range from conducting measurements, collecting and converting data from source formats to a common one supported by the common monitoring API, up to making data available for use. In addition, permissions have to be provided for users in order to observe or at least retrieve monitoring information about infrastructure resources used by their UCREs. Such a generic implementation of MAFIA concerns only the setup and deployment of monitoring services for users. To support the monitoring need of the federation services, namely SLA management and trustworthy reputation, specific monitoring resources need to be deployed. These resources are auto-configurable and capable of collecting monitoring data about pre-defined metrics related to UCREs during their lifetime and provide data to individual federation services. This is discussed in Sec. 4.3.2.3.

User-friendly Approach:

In contrast to the generic approach, MAFIA can be implemented in a user-friendly manner, where infrastructures offer monitoring service capabilities per offered resource type, and users can request the required resources and identify their interest in related monitoring services. For example, an infrastructure that offers resources of type VM might offer monitoring information about the created VMs (reflecting UCRE monitoring, see Sec. 4.3.1.3) and their hosting physical machines (reflecting infrastructure resources monitoring, see Sec. 4.3.1.2). Such capabilities can be advertised in the description of the VM resource type. A user can request one or multiple instances of the resource type VM and identify whether or not they are interested in monitoring information. From this perspective, infrastructures will then be in charge of setting up monitoring services and delivering the respective monitoring data to the user through the *common monitoring API* during the UCRE lifetime. Such setup and configuration per UCRE basis is analogue to the way data is provided to the federation services in the generic approach above. Hence, different setups and configurations are expected, as users might have different UCRE resources with different lifetimes and varying

interests in monitoring services. These are managed through different monitoring resource adapters and data wrappers as illustrated in Figure 4.6.

Having user-friendliness as the focus of this approach – in particular by reducing users' involvement in the setup and configuration – allows the following:

- Automatic setup of monitoring services
- Automatic adjustment to dynamic change of the deployed service or application resources (experiment resources within the context of the experimentation facilities)
- Easy access to monitoring data
- Providing data access during the lifetime of the service or application (the experiment within the context of the experimentation facilities) and afterwards for post-processing
- Providing data in unified data representation

The steps involved in the entire instrumentation process are illustrated in the sequence diagram in Figure 4.8 that is described as follows:

1. Infrastructure provider describes the offered resources associated with monitoring capabilities. As an example from a cloud-based infrastructure, a VM resource can be described along with an announcement of the capability of providing information about the VM (e.g. CPU-, memory-, disk-usage), its surrounding conditions (e.g. network performance) and its hosting physical machine (e.g. CPU-, memory-, disk-usage). It is left to the infrastructure provider to decide on the kind and frequency of the provided monitoring information.
2. Using any user tool provided in the federation, users get or discover the offered resources after being authenticated.
3. A user will then indicate interest in monitoring services while requesting resources. The user then configures the requests by identifying (through a unique ID) an endpoint where monitoring data should be pushed to. This endpoint can be the Uniform Resource Identifier (URI) of the user data collector. Such a collector can be deployed prior to requesting the resources used by the UCRE in order to collect monitoring data.
4. Upon receipt of a request, the infrastructure manager checks if monitoring services related to the requested resources are requested. If yes, a suitable wrapper resource is then deployed and configured to periodically fetch measurement data from the used monitoring tools, convert the data into the common data format and push it to the given endpoint of the collection resource. It is left to the

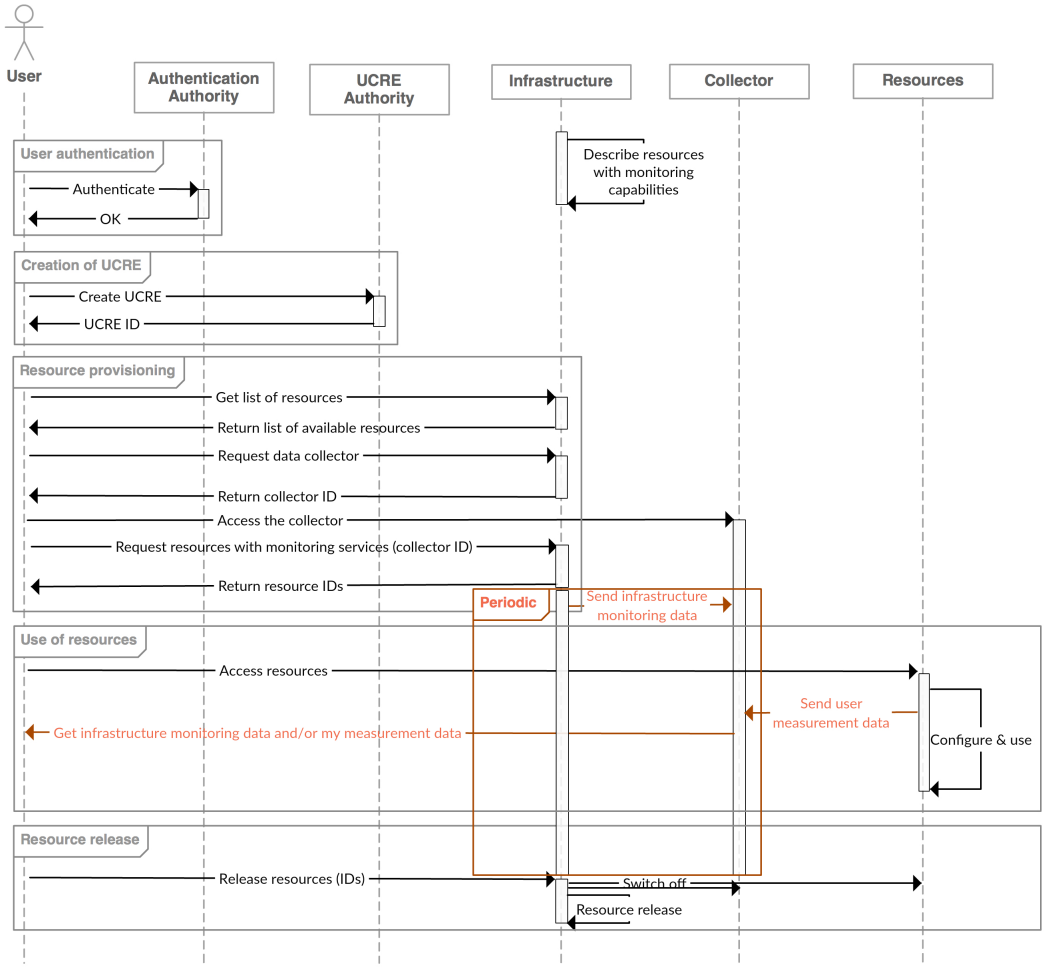


Fig. 4.8.: Sequence diagram for the setup of monitoring services in the user-friendly approach

infrastructure provider to use a single wrapper per resource type, per UCRE or per user. A wrapper resource is dynamically able to react to updates, e.g. when more resources are created or deleted.

5. Wrappers are released once the UCRE terminates.

The user-friendly approach focuses on providing an automatic setup and configuration of monitoring services that are grouped under UCRE monitoring and infrastructure resources monitoring. Services and application monitoring can be executed by users themselves. The rationale for this decision is that the behavior of the users' applications and services are only known to the users. They can deploy, configure and control any tools suitable to measure and monitor their running services or applications. Infrastructures could provide such tools in the form of software or deployable images with certain solutions pre-installed.

4.3.2.2. Infrastructure Health and Status Monitoring for Federation Administrators and FLS Monitoring Dashboard

ICT infrastructures are usually under operational monitoring and control of their administrators. To this end, low level infrastructure monitoring is supported at infrastructure level for internal administrative demands. For this purpose, various tools are used to monitor infrastructure resources and services. It's left to each infrastructure provider to deploy suitable tools depending on the type and size of the infrastructure.

These tools are also used to provide the required high level health and status information about the infrastructure to its consumers (federation and system administrators or FLS). The difference here, compared to the low level monitoring service, is that only a particular kind of information about the status of the key components in the infrastructure is required. That means specific metrics are measured that indicate the health and availability status of the infrastructure, the results of which are then reported. This information is made available via the common API. Each infrastructure can use a wrapper that is in charge of collecting the concerned data from the local monitoring tools and exporting the data to consumers (e.g. FLS monitoring dashboard as one of the federation services) following the rules and formats of the common API, as shown in Figure 4.6.

4.3.2.3. Infrastructure Resources Monitoring for Federation Services

Information about infrastructure resources, such as their availability, usage and performance, is required by multiple federation services as part of their functionalities.

Federation services have different requirements for monitoring information. Therefore, the types and frequency of the measured metrics for each service are to be identified and arranged between the individual infrastructure providers and the service developers. The heterogeneity of the federated infrastructures would be expected to lead to different sets of metrics to be measured for each federation service.

Furthermore, these services don't only require different kinds of information but also differ in delivery time intervals. For instance, in the case of the SLA management service, upon receipt of a request to create UCRE resources, the infrastructure manager is in charge of configuring proper software to provide related monitoring data of predefined metrics per UCRE basis only during its lifetime. While some other services, such as reservation or brokerage, require historical resource characteristics.

However, the common feature across all services is that monitoring data is exported by infrastructure providers through the common API to collection endpoints. For each federation service, one or multiple instances of a monitoring collection resource can be deployed. The number of instances deployed depends on the deployment of the federation service itself, as well as on the operational level of support offered by

infrastructure providers. If a federation service is deployed in a distributed manner (an instance runs in each infrastructure), it's recommended to deploy the collection resource at the infrastructure level. Additional reasons to deploy the collection resource at infrastructure level include reducing complexity and resource consumption (e.g. bandwidth), avoiding possible connectivity issues, and reducing latency, (Req-14, Req-16). If a federation service is deployed in a central manner, the deployment of the collector resource can either be deployed in a distributed manner, with one instance at each infrastructure, or one central collector at the federation level to be used by all infrastructures. The decision is left to be agreed upon between service providers and infrastructure providers.

4.4. Summary

In this chapter, the architecture design is discussed. The design went through an iterative and incremental process, as discussed in the introduction of this chapter. The conceptual phase of architecture design as discussed in Sec. 4.1 includes initial and final design phases. The initial phase focused on cloud federation (Sec. 4.1.1) and introduced an initial architecture (Sec. 4.1.1.3) that followed a set of architectural principles (Sec. 4.1.1.1). Decisions were taken to enhance the initial architecture design (Sec. 4.1.2) towards a more generic and flexible architecture that can be adopted in a heterogeneous federation in an extensible manner (Sec. 4.2), considering additional principles (Sec. 4.2.1). All principles take into consideration the fulfillment of the requirements identified in Sec. 3.2.

The final architecture, called (MAFIA), is introduced in Sec. 4.3. Its functional elements are described along with the various supported monitoring services (Sec. 4.3.1), some of which were already covered in the initial design (Sec. 4.1.1.2). Various sets of datasets are collected from different sources across the federation and provided for multiple, various stakeholders through a common API.

The most suitable candidate for a reference implementation of the common API has to allow representing and transporting data in a common format, as well as using flexible, user-defined schemas. However, providing data through a common format may cause confusion if all parties using the interface don't have the same understanding and definitions for the used concepts and their relations. Such flexibility, if own vocabularies are used, leads to an interoperability issue. This can be treated at information model level through defining a common, shared information model to represent monitoring data, concepts and relationships in a unified, standardized manner following a shared vocabulary.

Therefore Chapter 5 introduces an information model by leveraging Semantic Web technologies.

5 Ontology-Based Information Model

THIS chapter presents the common information model designed and implemented within this thesis. This model serves and supports compatibility and interoperability of tools and components across federated ICT infrastructures in terms of monitoring data exchange, as per the requirements identified in Chapter 3 (Req-8, Req-11 and Req-22). The model is used in the implementation of the components of the monitoring architecture (MAFIA, presented in Sec. 4.3), in order to allow representing and transporting monitoring data through its common API using unified, meaningful, shared and standardized vocabulary. This allows the unification of the data schemas coming from different sources with similar purposes to facilitate tools' interoperability.

The state-of-the-art data models as discussed in Sec. 2.6.2 are limited to the network measurement domain and are implemented for specific protocols or frameworks,

which makes their use very complex in a broader context. They have different concepts, schemas and modeling mechanisms, as they were designed based on diverse requirements and applied contexts. Such domain-specific and implementation-oriented models lead to poor, limited interoperability and wasted efforts reinventing the same subject matter.

A feasible way to have a generic, reusable, widely used model is to reduce or eliminate conceptual and terminological confusion through a common, shared understanding and definitions of concepts and their relationships. This will serve as the basis for interoperability of systems and data, as well as for reusability and reliability [109].

For this purpose, ontology-based modeling of the Semantic Web has tremendous potential as a valid approach for developing a common, extensible information model. Ontologies by definition have to be as generic and task-independent as possible [154].

The following sections present the main concept, as well as introduce and describe an ontology-based information model, called Monitoring Ontology for Federated Infrastructures [109], [159] (MOFI).

5.1. Main Concept of Ontology-Based Modeling

This section gives an overview of the main concepts of semantic ontologies and their advantages.

An ontology defines a formal, explicit vocabulary and definitions of shared concepts and their relationships within a given domain [155]. As this thesis focuses on the monitoring domain, the MOFI ontology includes formal vocabulary covering related concepts like measurement metrics, data, data units, and their relationships in an explicit (representative and meaningful) manner.

Ontologies enable formally defining concepts and their relationships in taxonomies (superclass–subclass hierarchies) including rich semantic meanings. Classes refer to concepts and each class represents group of individuals that share common characteristics. Concepts usually have properties that represent their relations. Ontologies enable describing information at different levels of abstraction, i.e. allowing the definition of specific classes derived from generic ones. For example, the class `CPULoad` has a data property called `hasMeasurementData`, which has `Data` as a range, as well as an object property called `isMeasuredBy`, whose range is a measurement tool (e.g. `Zabbix`), which is an individual of the class `MonitoringTool` that is a subclass of the generic class `Tool`.

The strengths of ontology-based modeling are its focus on interoperability, schema unboundness, extensibility and reusability [127]. Using formal, explicit and shared vocabulary ensures compatibility and interoperability of tools. Ontologies are extensible to include additional vocabularies, and their vocabularies are reusable by other ontologies. The extensibility feature facilitates smooth extensions and the integration

of further segments in the systems and architectures that implement these ontologies. The reusability feature facilitates the sustainability of the ontology-based models and reducing the effort of others as they don't need to re-design the same subject matter. Finally, dealing with information at a semantic level is recognized as a powerful solution, as it enables, besides representing data in a common manner, some degree of inference and automatic reasoning over the concerned monitoring data [156]. More details on ontology-based modeling are given in Sec. 2.6.4.

5.2. MOFI: Monitoring Ontology for Federated Infrastructures

The development of MOFI is driven by the practical use and requirements of MAFIA, which focuses on monitoring federated infrastructures. More precisely, MOFI serves to share a common understanding of the structure of information among people or software architectural elements in the domain of monitoring, which aligns with the main and common goals in developing ontologies [124], [157]. The MOFI ontology is described after introducing the design decisions taken prior and during its development.

5.2.1. Design Decisions

MOFI is used as a common information model implemented within MAFIA. It covers various monitoring services provided for different stakeholders: federation administrators, experimenters, federation services (e.g. SLA management, trustworthy reputation, reservation) and the FLS monitoring dashboard, as discussed in Sec. 4.3. MAFIA assumes a common API is used for data provisioning across the federation.

As stated in Sec. 4.4, providing data in a common format through a common API without following a common information model only helps solve part of the problem addressed in this thesis. MOFI is therefore required in order for all parties involved in MAFIA to have the same understanding and definitions for the concepts used and their relations, and thus, to eliminate any possible confusion or misinterpretation.

For developing an ontology, a proper methodology should be used [158]. The top-down method is followed, as described below:

- The area of interest was defined, namely monitoring services and measurement metrics in federated, heterogeneous ICT infrastructures. The scope of the ontology is limited to the common requirements and services covered by MAFIA, taking Fed4FIRE as a reference federation.
- The important concepts and their relations were identified.
- Existing ontologies were reused and extended as much as possible to cover various domains in large ICT federations. Concepts and relations have been reused from

the World Wide Web Consortium (W3C) XML Schema Definition (XSD)[115] and W3C Time[1]. MOMENT and NOVI are taken as a starting point, which in turn considered NM-WG, PerfSONAR and NASA Units[2] models.

- Concepts were defined as classes and arranged in a taxonomic hierarchy following a top-down method. Definitions of the most general concepts were defined first and specialization of the concepts after that.

- Concepts' relations were defined as object and data properties, along with rules for their domains, ranges and restrictions.

- Relevant individuals (instances) of the defined classes were created.

For MOFI to be reused beyond its current use and for standardization aims, it is designed and developed together with other ontologies in order to model different aspects within federated infrastructures. Each of these ontologies focuses on a particular domain, i.e. modeling federation concepts, infrastructures, services, resources, components, policies and further domain specific concepts such as wireless, cloud, SDN, etc. MOFI together with all these ontologies form the Open-Multinet[3] (OMN) Ontology [129], which focuses on describing federated infrastructures and their resources, as well as supporting management of the whole ICT experimentation lifecycle. The OMN ontology defines in its upper layer ontology (with the namespace prefix *omn*) several basic concepts and their relations that are then reused and specialized in the subjacent ontologies (*omn-component*, *omn-resource*, *omn-service*, *omn-federation*, *omn-lifecycle*, *omn-monitoring*, *omn-policy* and others). MOFI represents the *omn-monitoring* ontology and is then directly linked to other *omn* ontologies that are considered external ontologies to MOFI.

For better understanding the requirements of the desired information model, Figure 5.1 illustrates the main, relevant components and their interactions within MAFIA. Although some components are out of MOFI scope, they are represented in the diagram to shed light on the broader scope. However, these components, namely the infrastructure manager, federation services (e.g. SLA management, trustworthy reputation and reservation), resources, components, experiments and applications, and infrastructures, are modeled within the *OMN Ontology*. The focus of the MOFI information model is monitoring and measurement tools, measurement metrics, measurement data, data units and further related concepts. It should serve MAFIA's needs, and thus model all the components involved in the entire monitoring process. This allows the provisioning of different sets of monitoring information for various consumers, like the FLS monitoring dashboard, federation services, and users.

[1] http://www.w3.org/TR/owl-time/
[2] http://sweet.jpl.nasa.gov/ontology/units.owl
[3] http://open-multinet.info

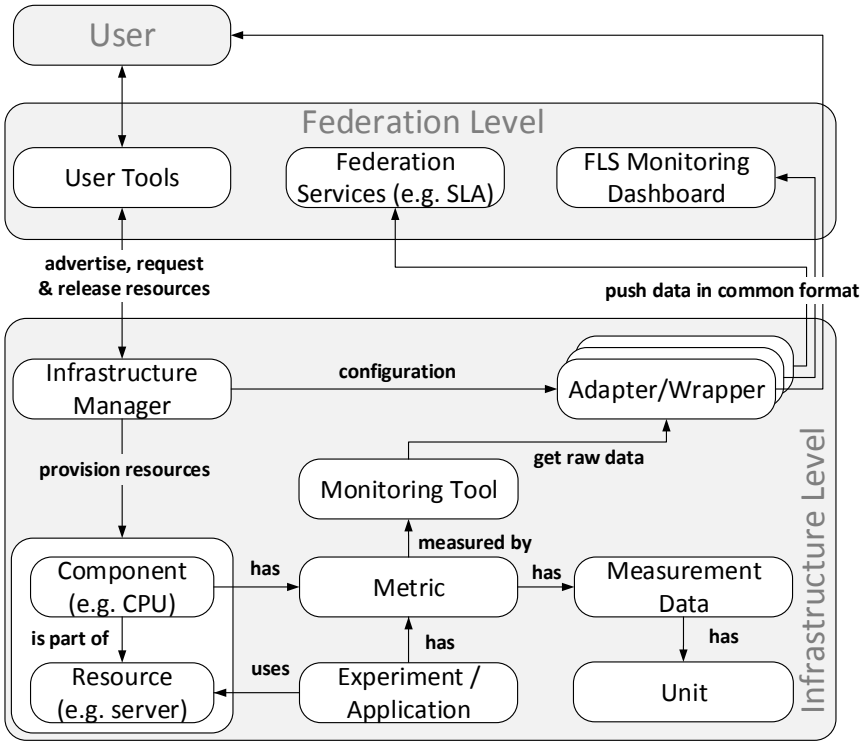


Fig. 5.1.: Concepts of interest for the target information model [109]

The management of large ontologies is a serious challenge for designers and users. Modularization is a valid strategy to deal with this problem, i.e. to divide the subject matter into smaller ontology components (sub-ontologies) [36]. MOFI is divided into a hierarchy of different ontologies as shown in Figure 5.2. The highest level (upper layer) ontology (with the namespace prefix *mofi*) describes very general concepts and properties that are reused and specialized in the subjacent ontologies (*mofi-metric*, *mofi-data*, *mofi-unit*, *mofi-tool*, *mofi-genericconcepts*). These ontologies are described in the following sections.

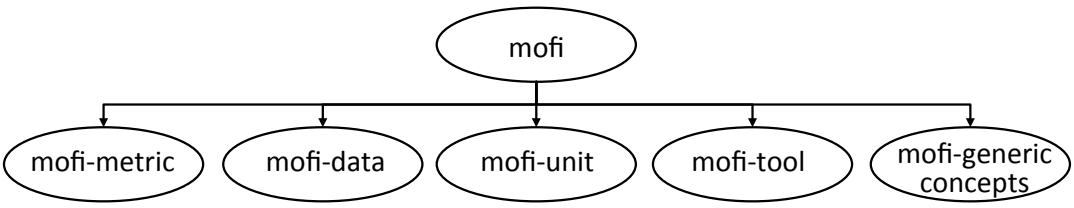


Fig. 5.2.: MOFI hierarchy

5.2.2. MOFI Upper Ontology

The basic and fundamental measurement and monitoring concepts and their relations are described in the upper ontology through a set of classes (e.g. `Metric`, `Data`, `Unit` and `Tool`) and properties (e.g. `measuredBy`, `measuresMetric`, `pushesDataTo`, `hasUnit` and `retrievedBy`). Furthermore, different sets of monitoring services in federation infrastructures are defined, such as `InfrastructureHealthMonitoring`, `InfrastructureResourceMonitoring`, `SLAMonitoring`. These are subclasses of the class `MonitoringService`, which is in turn a subclass of `omn:Service` like `Measurement` and `FirstLevelSupport` as shown in Figure 5.3. The figure shows only part of the ontology.

Interactions with these services are described through `isRequested`, `isOffered`, `requiresUsername`, `requiresPassword` data properties. Besides defining the relation between these classes, the upper ontology includes some properties that link these classes with other external ontologies. The `isMeasurementMetricOf` and its inverse property `hasMeasurementMetric` express the relations between `Metric` and `omn:Service`, `omn:Resource` and `omn:Component`, while the relation between `Tool` with these OMN classes can be expressed through the `monitors` property.

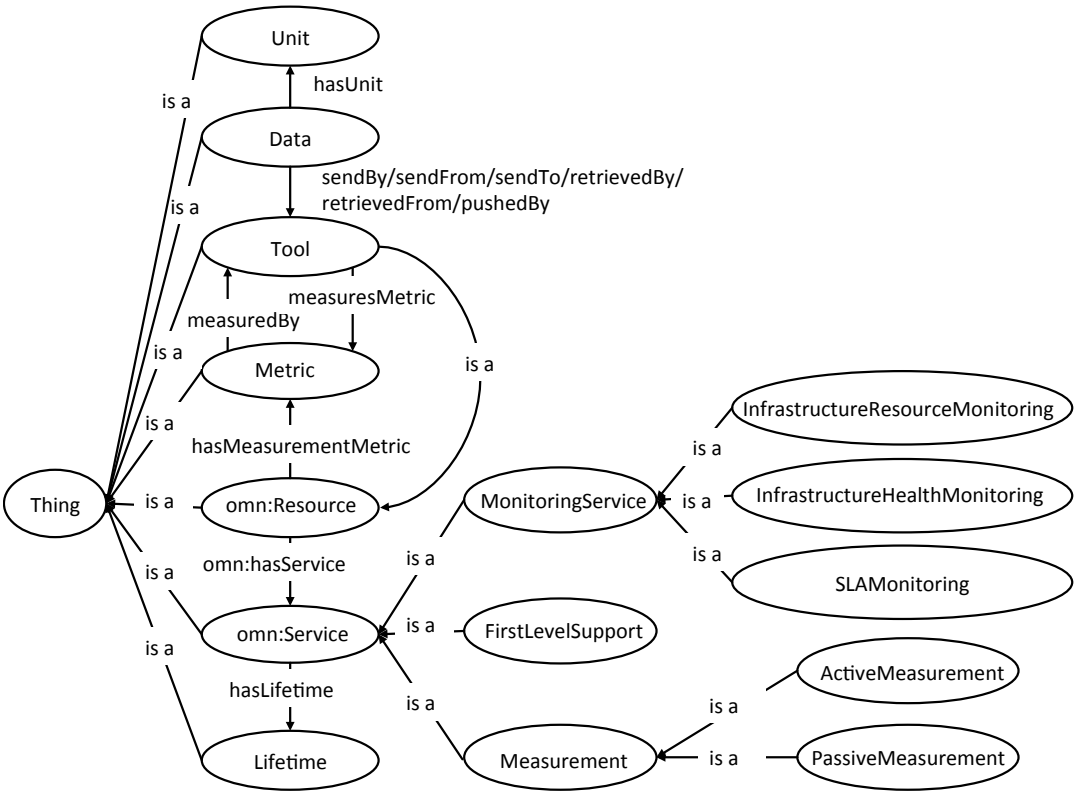


Fig. 5.3.: MOFI Upper ontology (based on [159])

5.2.3. MOFI Metric Ontology

The Metric ontology describes anything that can be measured and monitored in federated ICT infrastructures. Over fifty measurement metrics used for cross-layer monitoring in different domains (e.g. datacenters, wireless networks, IP traffic) are described as classes. These classes describe metrics whose information changes dynamically (e.g. CPU utilization, memory consumption, and packet delay and loss), as well as metrics whose information may change very infrequently over time (e.g. CPU core counts in a machine).

Figure 5.4 illustrates some classes that represent a set of metrics. Some classes include further subclasses. For instance the `RadioSignalQuality` class includes a set of subclasses representing radio signal quality in the wireless domain, such as radio signal strength indicator level (`RSSILevel`), `NoiseLevel`, signal to interference ration (`SIR`) and others. Figure 5.4 does not cover the whole taxonomy, but rather gives an idea of the model. For instance `MemoryUtilization` includes more subclasses than shown (e.g. `SharedMemory` and `BuffersMemory`), while other classes are further subclassified, like `Delay` (e.g. `OneWayDelay`, `RoundTripDelay`) and the `CPUUtilization` (e.g. `CPULoad` and `AllocatedCPU`).

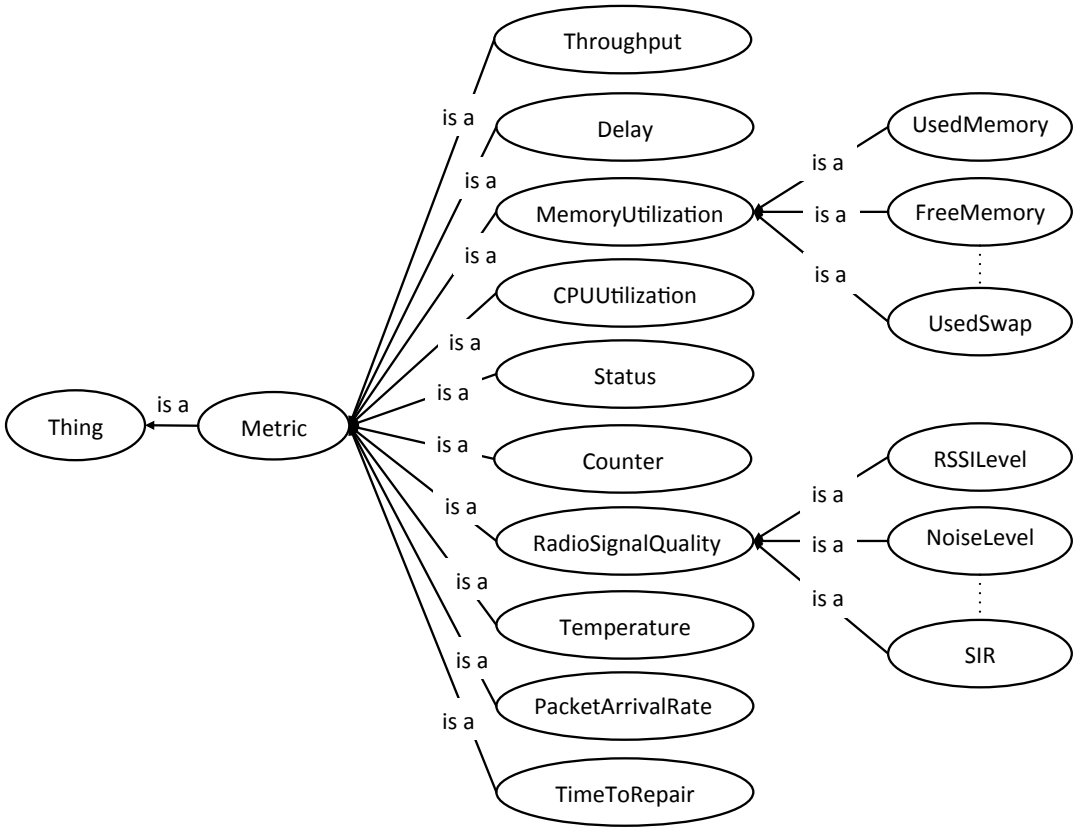


Fig. 5.4.: MOFI Metric ontology (based on [159])

In addition to the properties in the upper ontology that represent relations with the type `Metric`, further properties involving the class `Metric` are defined in the MOFI Metric ontology like `hasFrequency` and `statusValue`.

5.2.4. MOFI Data Ontology

The Data ontology describes basic concepts related to data. Figure 5.5 includes a number of those classes that represent different types of measurement data (e.g. `SimpleMeasurement`, `StatisticalMeasurement` and `MeasurementParameter`) and data formats (e.g. `DataFormat`, `FormattedFile` and `UnformattedFile`). These classes and others are linked through a set of object properties (e.g. `dataFormat`, `hasMeasurementData` and `isStatisticalMeasurementOf`) and data properties (e.g. `hasTimestamp` and `hasMeasurementDataValue`).

Several classes and properties defined in the MOMENT ontology have been reused in this ontology. However, some of the reused classes have been re-classified and some properties have different domains and ranges. This has been done in order for the MOFI Data ontology to be generic enough to serve heterogeneous (federated) domains, as MOMENT focused only on network measurement.

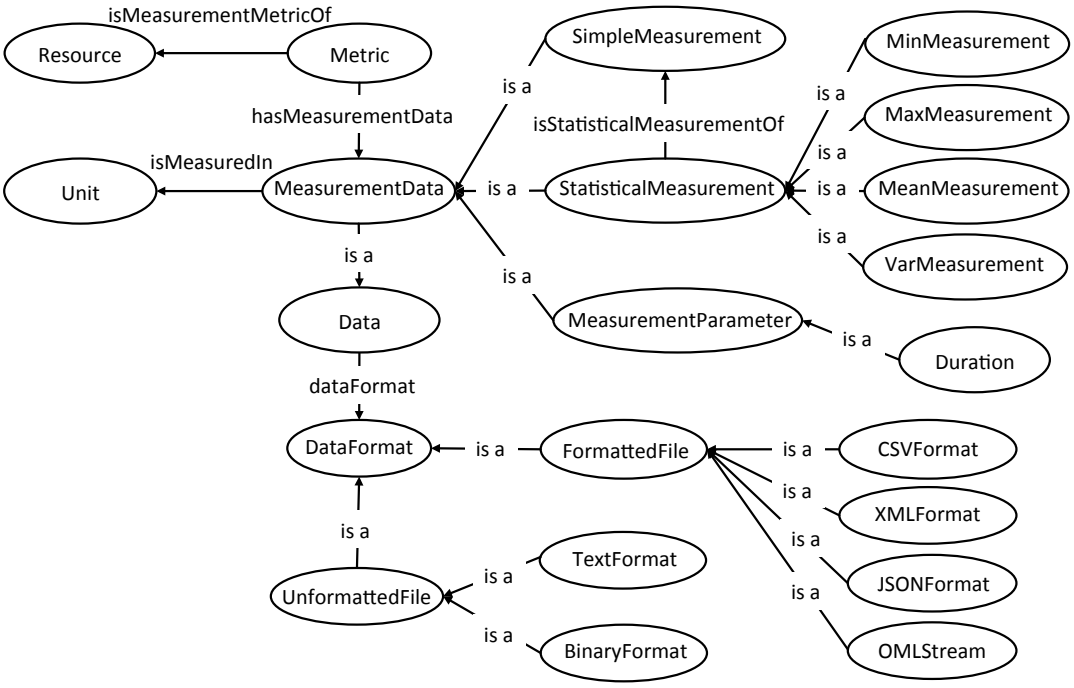


Fig. 5.5.: MOFI Data ontology (based on [159])

5.2.5. MOFI Unit Ontology

The Unit ontology developed by the European project NOVI has been completely reused because it serves the target purpose. The NOVI Unit ontology was initially defined within the European project MOMENT, which leverages the National Aeronautics and Space Administration (NASA) Unit ontology[4], but replaced the units from the physics field used by NASA with those from computer science like bit, byte, bit per second (bps), IP address, etc. [34].

The MOFI Unit ontology includes data units (e.g. bit, byte, second, bps) covering both unit prefixes (e.g. kilo, mega, giga), binary and decimal, as well as different measurement levels (nominal, ordinal, interval, and ratio) and dimensions (basic and derived) as illustrated in Figure 5.6.

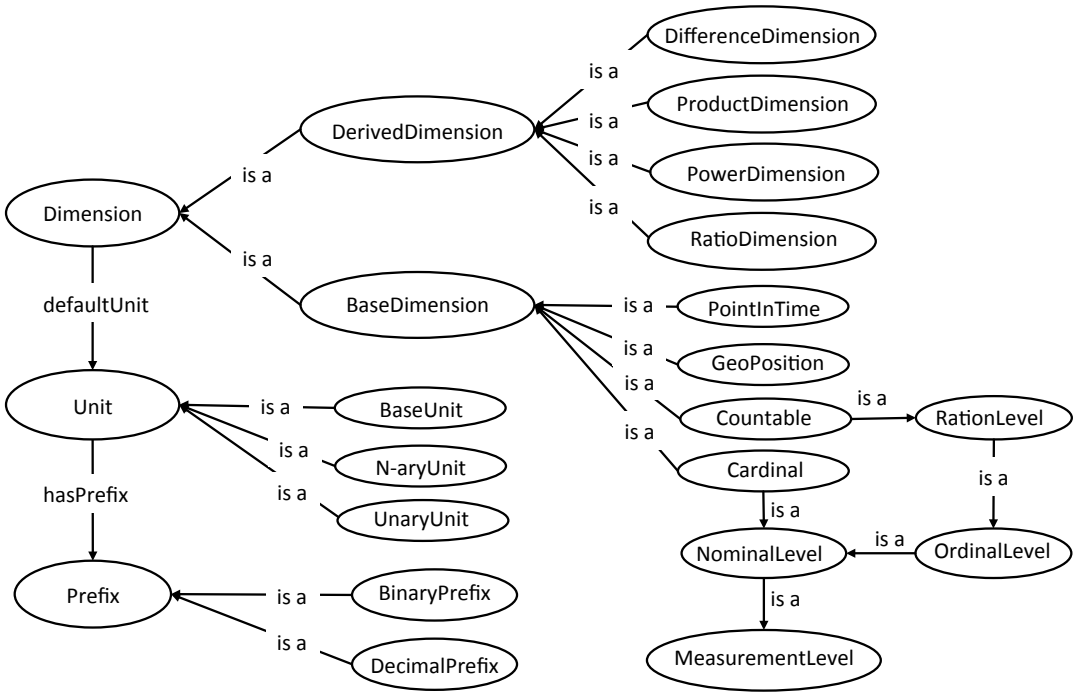


Fig. 5.6.: MOFI Unit ontology (based on [159])

5.2.6. MOFI Tool Ontology

The monitoring process goes through multiple stages: data acquisition, collection, storage, publication and visualization of the data, or even sending notifications or events. The data might be filtered, converted from one format to another or analyzed.

A wide range of measurement and monitoring tools exist that differ from one other in their core functionalities, architectures, communication paradigms and data

[4] http://sweet.jpl.nasa.gov/ontology/units.owl

access capabilities. These concepts are described in this ontology through classes (e.g. `MeasurementTool`, `CaptureTool`, `Collector`, `Adapter`, `VisualizationTool`, `CollectionEndpoint` and `Database`) and properties (e.g. `communicationParadigm`, `dataAccessProvided` and `reportsDataAbout`), as partially shown in Figure 5.7.

The common tools that are used in the Fed4FIRE federation for instance are defined as individuals (e.g. `Zabbix`, `Nagios`, `Iperf`, `ping`, `Traceroute`, `OMLServer`, `OMLWrapper`, `PostgreSQL`, `JenaFuseki`, `OMSPEndpoint`, and `SPARQLEndpoint`). Some concepts defined in the MOMENT ontology have been reused, e.g. subclasses of `MonitoringTool` and `CommunicationParadigm`. The figure shows only part of the ontology.

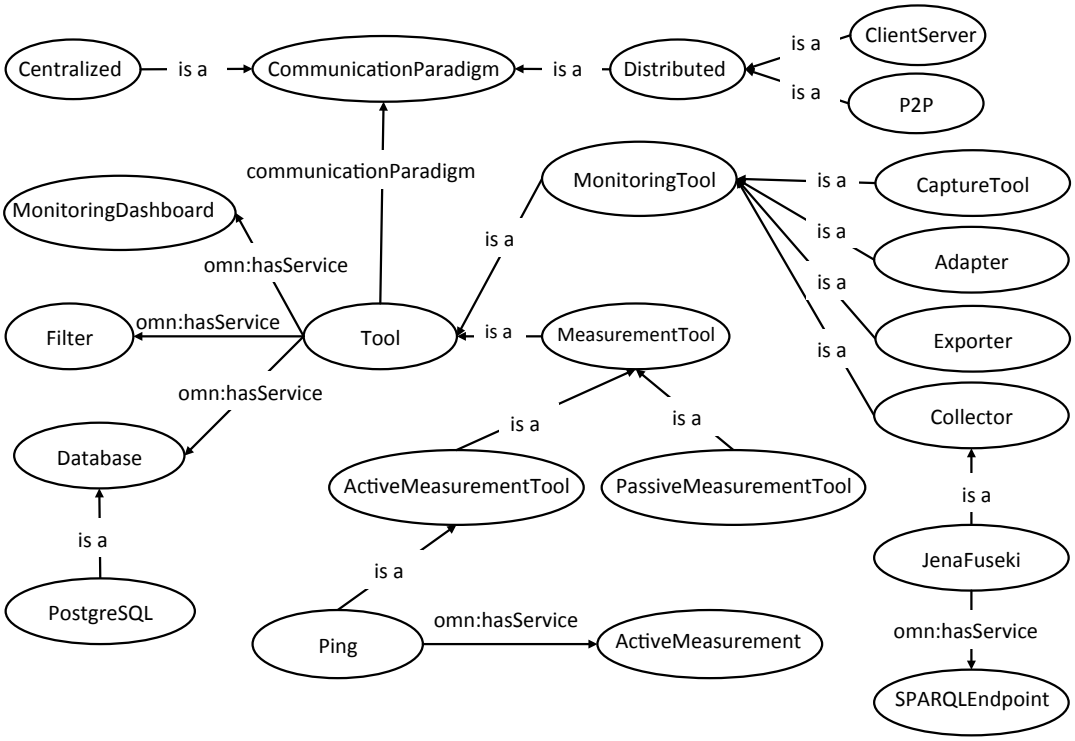


Fig. 5.7.: MOFI Tool ontology (based on [159])

5.2.7. MOFI Generic Concepts Ontology

The Generic Concepts ontology describes generic concepts and relations that are important for measurements, monitoring services and tools, but which were not described in the previous ontologies. It covers a set of classes describing concepts, such as `Location`, `LogicalLocation`, `PhysicalLocation`, `Protocol`, `Event`, `Query` and `AuthenticationMethod`. It also includes a couple of properties describing relations, like `locatedAt`, `latitude`, `longitude`, `usesProtocol`, etc.

5.2.8. Interaction with External Ontologies

MOFI is directly linked to other external ontologies that are part of the OMN ontology (like *omn*, *omn-resource*, and *omn-service*).

An example of the interactions of MOFI subontologies with external ontologies is illustrated in Figure 5.8.

The external ontologies and their relations with each other are represented with dotted lines. To facilitate understanding of Figure 5.8, imagine the following scenario. A user has created a server (modeled as `omn:Resource`) in a particular infrastructure. Monitoring data is needed about the usage of the server for the SLA management service (modeled in the *omn-service* ontology as a subclass of `omn:Service`). Measurement data (modeled in *mofi-data* as a subclass of `mofi:Data`) with the proper units (modeled under `mofi:Unit`) about some relevant metrics are provided through an OML wrapper (modeled in *mofi-tool* ontology as an individual of type `mofi:Tool`) to the SLA service. An example of such a metrics is load (modeled in *mofi-metric* as a subclass of `mofi:Metric`) of the CPU component (modeled as `omn:Component`), which is part of the server resource.

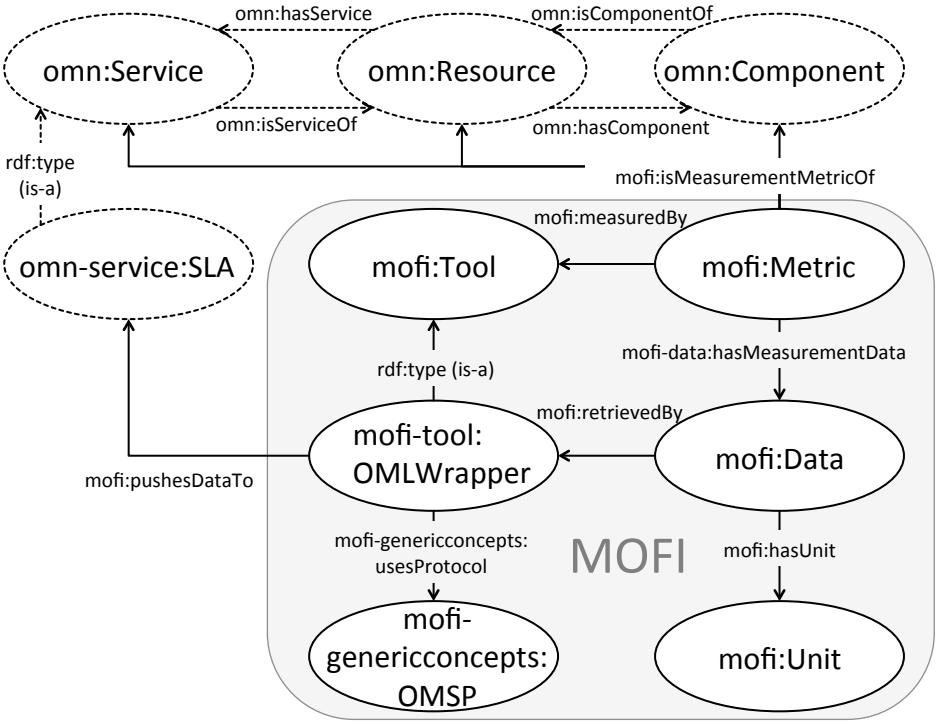


Fig. 5.8.: Example illustrating MOFI interactions with external ontologies (based on [109])

5.2.9. Data Modeling and Serialization

OWL is used to describe the MOFI ontology and together with RDF and RDFS represent the data model. RDF/XML and RDF-Turtle are used for data serialization. Parts of MOFI ontologies, written in Turtle, which is a human and machine readable language, are given in Appendix B. The open source ontology editor Protégé[5] was used to define the ontology.

5.3. Summary

This chapter introduced the MOFI ontology, which acts as a common information model for MAFIA. After introducing the main concept behind using ontologies in this thesis in Sec. 5.1, in Sec. 5.2, MOFI with its six subontologies are described. The formal description of MOFI includes over 1500 triples covering the common and mostly used concepts and relations in the domains addressed within this thesis. Its scope is limited to the requirements of MAFIA considering one of the largest ICT infrastructure federations in Europe, Fed4FIRE, as a reference in order to define the common concerned aspects. The methodology and design decisions are discussed in Sec. 5.2.1.

The introduced ontology models are implemented within MAFIA, whose implementation is discussed in Chapter 6.

[5] http://protege.stanford.edu

6 Implementation of the Architectural Functional Elements

THIS chapter presents a reference implementation of the monitoring architecture (MAFIA) and the information model (MOFI) that are designed within this thesis as presented in Chapter 4 and Chapter 5 respectively. As mentioned in Chapter 4, the architecture design was done in an iterative and incremental manner and therefore went through two main phases. Accordingly, the implementation steps were also done in iterations during these two phases.

The goal of this chapter is to introduce a reference implementation of MAFIA. Note that the strategy followed in this thesis, including the implementation, is to reuse existing technologies, protocols and tools as much as possible as long as they serve the required functionalities. Therefore, only functional elements of the architecture that are neither supported by the state-of-the-art solutions nor in line with the defined design principles are implemented.

Furthermore, the reference implementation provided by this thesis covers the monitoring of some domains (e.g. clouds and cellular networks) in a federation. However, it is extensible to cover monitoring of any ICT domain or specific application area.

The focus of this chapter is the implementation of the final architecture design. However, the implementation of the initial design is first briefly described. This is because the final design is based on the architectural design principles of the initial design. Furthermore, the experience gained after the implementation and deployment of the initial architecture deign (Sec. 4.1.1.4) leads to the demand of a new design as discussed in Sec. 4.1.2.

6.1. Overview of the Implementation of the Initial Architecture

Offering on-demand cross-layer and cross-domain monitoring services for users was at the heart of the initial design. The goal was to support monitoring of heterogenous resources deployed across federated cloud infrastructures. That means there was no focus on heterogenous domain infrastructures, as all federated infrastructures were cloud-based, which are heterogenous in terms of structure, networking features and resources, and the used cloud management solutions.

This design follows the set of architectural design principles discussed in Sec. 4.1.1. It assumes the use of a homogenous monitoring solution across the federation. In its implementation, Zabbix (see Sec. 2.5.2.4) is adopted as a suitable monitoring solution that fulfills the requirements of the initial architecture design. The choice of Zabbix was made after comparing several alternative existing monitoring tools (Zabbix, Nagios, OpenNMS, Collectd and Ganglia).[1] These were compared in accordance to a set of features that should be supported, like distributed monitoring, auto-registration of monitoring clients, SNMP support, triggers and alerts, flexibility of data storage and the support of various databases, access control, support of screens and maps and Internet Protocol version 6 (IPv6) support. Not all but most of these features are supported by all the mentioned tools. However, besides supporting all these features, Zabbix has essential advantages for the target architecture, such as ease of deployment and customizability.

Monitoring functionality is implemented in a simple, traditional manner following a client-server model [64], as shown in Figure 4.3. As this section focuses only on the implementation of monitoring components, the implementations of other components in the architecture, such as the Web portal, resource manager and cloud infrastructure manager, are not presented. Nevertheless, the use of these reference implementations within the context of their use within MAFIA are discussed later in Sec. 7.1.

Monitoring services are offered for users on an on-demand basis. If a user is interested in monitoring, they have to create a monitoring collector resource (referred to as a User Collector in the design) in the form of a VM that uses a special image

[1] These tools are presented in Sec. 2.5.2.1

referred to as *collector image*. Each user has a dedicated collector for exclusive use that is deployed as a separate resource and collects monitoring data reported by clients (called agents in Zabbix) that reside in the UCRE's VM images. Amongst the types of agents that Zabbix supports (see Sec. 2.5.2.4), the active Zabbix agent mode is used in this implementation to avoid possible accessibility problems because of Network Address Translation (NAT). Using this mode, the agent is the one that initiates communication and sends monitoring data to the server. However, the Zabbix server is configured to enable its auto-registration feature in order to allow Zabbix agents running anywhere to be automatically registered once they are configured with the server IP. Furthermore, in order for agents to report the required data, a set of basic measurement metrics (like CPU usage, memory and disk consumption, and more) are preconfigured through a template stored in the server. This template is then used by all agents to report data values for these metrics. Linking this template to each agent is done automatically if the auto-registration feature is enabled and linked to the template.

The collector VM is created like any cloud compute resource (VM) through the Web portal or directly through an OCCI-based API exposed by the resource manager. This manager maps and passes user requests to the cloud infrastructure manager (e.g. OpenNebula[12] or OpenStack[11]) of the respective cloud infrastructure, as illustrated in Figure 4.3. The Web portal allows the creation of cloud resources (VMs, storage, etc.) through a GUI. It allows users to design their UCREs and sends XML-based OCCI requests to the resource manager. While creating the collector, the user can choose where to store the data, either inside the collector VM or in an external storage resource. This is implemented in a way to allow the user to identify an option and, accordingly, the database will either be stored inside the collector image or in an external storage resource that is attached to the collector resource as an additional, external disk. Such capability enables the user to store the data in on-demand storage space that might be permanently available, and to have the possibility of reusing an external storage resource in the future for post-processing or data analytics. The user's choice is passed to the collector through the *contextualization service* supported by OCCI and the cloud management software (OpenNebula is used in this implementation). Contextualization information is checked while booting the collector VM and proper configuration actions are taken accordingly.

The user collector has an IP address that is reachable by any VM deployed anywhere across the federation. This is because all the federated infrastructures, where this implementation is deployed, are located within a single WAN that facilitates communication between VMs located at different infrastructures. Each infrastructure is connected to every other one via a direct VPN connection. Upon creation of the collector, its IP is used by all user VMs that are monitored.

6.1.1. Main Monitoring Components

6.1.1.1. Monitoring Collector Image

The collector image contains both Zabbix server and agent, as well as further preinstalled initialization scripts. The initialization scripts are used to deploy the requested monitoring services. These services include infrastructure resources monitoring, the use of permanent external storage, VM event logs, and allowing the collector to monitor itself. The initialization scripts are configured according to contextualization information received. This information includes users' interest in any of the monitoring services and is sent along with the collector creation request.

The federation portal used in the implementation supports the automatic creation of the collector VM. Once the collector is running, the user can browse its Web GUI or directly invoke its JSON-RPC API to get data in JSON format. Beyond the native JSON-RPC API supported by Zabbix, a script was implemented to export data in CSV format. Furthermore, a Representational State Transfer (REST) API was implemented that provides data in JSON format over HTTP.

Each monitored machine is referred to in Zabbix as a host and is stored in the Zabbix server with its original hostname. The collector image is prepared in a way so as to group all the UCRE's VMs under one hostgroup and all Physical Machines (PMs) that host these VMs under another hostgroup.

Requirements At least one cloud infrastructure has to provide a collector image that has a predefined ID. Such an ID may change during the lifetime of the infrastructure. Therefore, before triggering the OCCI request to create the compute resource running the collector, one must identify the image ID. In the case where all cloud infrastructures provide collector images, the choice of where to deploy the collector is left to the user. In this kind of implementation, the images used contains Zabbix version 1.8.6, which in turn uses MySQL[2] 5.0+, Apache[3] 1.3.12+ and PHP: Hypertext Preprocessor (PHP)[4] 5.1.6+.

Installation The OCCI request shown in Listing 6.1, when sent to the resource manager, triggers the creation of the collector image in the specified cloud infrastructure.

Using the `usage` element of the `context` tag of the OCCI request to identify required monitoring services (discussed in Sec. 6.1.2), the collector will automatically be configured accordingly. If external storage is required, prior to the creation of the collector, a storage resource (of type `datablock`) has to be deployed. Its ID is then used in the OCCI creation request of the collector as shown in Listing 6.2. The user

[2] https://www.mysql.com
[3] http://httpd.apache.org
[4] http://php.net

still needs to specify their desire to use external storage via the contextualization service with the `usage` element. This external storage will be mounted as an additional disk beside the main one used by the OS image as shown in Listing 6.1.

Listing 6.1: OCCI request to create a monitoring collector

```
$ curl -kni https://federation-API-entrypoint_ip/UCRE_id/computes \
-X POST -H'Content-Type: application/vnd.MediaType+xml' \
-d '<?xml version="1.0" encoding="UTF-8"?>
<compute xmlns="http://federation-API-entrypoint_ip/doc/schemas/occi">
  <name>MonitoringCollector</name>
  <instance_type>small</instance_type>
  <disk>
    <storage href="/locations/CloudInfrastructureName/storages/image_id"/>
    <type>OS</type>
    <target>hda</target>
  </disk>
  <nic>
    <network href="/locations/CloudInfrastructureName/networks/network_id"/>
  </nic>
  <context>
    <usage>zabbix-agent</usage>
  </context>
  <link href="/locations/CloudInfrastructureName" rel="location"/>
</compute>'
```

Listing 6.2: Part of an OCCI request to create a monitoring collector

```
  ...
  <disk>
    <storage href="/locations/CloudInfrastructureName/storages/datablock_id"/>
    <type>disk</type>
    <target>hdb</target>
  </disk>
  ...
  <context>
    <usage>zabbix-agent;zabbix-external-storage</usage>
  </context>
  ...
```

6.1.1.2. Compute Resource Image

Each federated cloud infrastructure may provide a variety of images for booting the offered compute resources (classic VMs). Images vary from each other in their OS or even in services offered (e.g. VNFs). To support monitoring, each of these images contains an instance of Zabbix agent software (all versions are supported). The image is preconfigured to utilize the contextualization information to activate the Zabbix agent if monitoring is required and configure it with the IP of the collector. In addition to those metrics predefined in the Zabbix template, the federation resource manager's OCCI API supports specification of monitoring metrics in the contextualization section of a compute resource [64]. Thus, users are able to define custom metrics when

preparing their VM images. The images include software (`init.sh` script) to read this contextualization information (stored by the cloud manager, such as OpenNebula, in a `context.sh` script) and configure the agents accordingly. Furthermore, users have the ability to specify additional monitoring metrics for already running VMs through additional software written in Python. This software takes as input a metric name, the Unix command used to produce the measures, and additional metric attributes like the update rate of measurement data in seconds, the amount of time to keep data history in days and the expected data type of the measures. To activate measuring for new metrics, the software automatically configures the local Zabbix agent and, through the use of the `zabbix_api.py` library, it configures the Zabbix server remotely using its JSON-RPC API. This service allows users to customize and monitor their services and applications.

Requirements Similar to the collector image, the user needs to identify the allocated ID of the selected image before the installation is performed.

Installation The OCCI request shown in Listing 6.3, when sent to the resource manager, triggers the creation of a VM in the specified cloud infrastructure.

Listing 6.3: OCCI request to create a VM with monitoring service enabled

```
$ curl -kni https://federation-API-entrypoint_ip/UCRE_id/computes \
-X POST -H'Content-Type:␣application/vnd.MediaType+xml' \
-d '<?xml␣version="1.0"␣encoding="UTF-8"?>
<compute␣xmlns="http://federation-API-entrypoint_ip/doc/schemas/occi">
␣␣<name>ComputeResource</name>
␣␣<instance_type>lite</instance_type>
␣␣<disk>
␣␣␣␣<storage␣href="/locations/CloudInfrastructureName/storages/image_id"/>
␣␣␣␣<type>OS</type>
␣␣␣␣<target>hda</target>
␣␣</disk>
␣␣<nic>
␣␣␣␣<network␣href="/locations/CloudInfrastructureName/networks/network_id"/>
␣␣</nic>
␣␣<context>
␣␣␣␣<collector_ip>MonitoringCollector-IP</collector_ip>
␣␣␣␣<usage>zabbix-agent</usage>
␣␣␣␣<metrics>
␣␣␣␣␣␣<metric>metric_name,␣Unix_command,␣metric_attributes</metric>
␣␣␣␣␣␣...
␣␣␣␣</metrics>
␣␣</context>
␣␣<link␣href="/locations/CloudInfrastructureName"␣rel="location"/>
</compute>'
```

Besides custom metrics defined by the user, the IP address of the collector is passed to the VM using the contextualization service supported by the OCCI creation request.

6.1.1.3. Contextualization Service

Contextualization is used to allow the automatic installation of any software on a VM after booting. It also allows parameters to be passed directly to the VM on instantiation. It is supported in both image types: collector and compute resource.

Collector Image Through the OCCI creation request (Listing 6.1), the user can identify the software to be installed after booting. This is supported through setting the `usage` element of the `context` tag by identifying the scripts to be executed. Listing 6.4 shows the possible setting of the `usage` element of the *context* tag of the OCCI request to create a collector VM. It includes four values. The first two have been already discussed, while the last two will be discussed in Sec. 6.1.2.

Listing 6.4: Use of the `usage` element of the OCCI `context` tag to create a collector VM

```
1 ...
2 <context>
3   <usage>zabbix-agent;zabbix-external-storage;log-MQevents-in-zabbix;infra-monitoring-
        init</usage>
4 </context>
5 ...
```

Compute Resource Images Through the OCCI creation request (Listing 6.3), the user can identify whether the created VM needs to be monitored. This is done through setting the `usage` element of the *context* tag to `zabbix-agent`, which will configure and run the Zabbix agent on the VM after booting. Configuration information such as the IP address of the collector and custom metrics, is also passed to the compute resource VM through the `context` tag of the OCCI request.

6.1.2. Cross-Layer Monitoring Support

Multiple monitoring services are supported. These are offered on-demand. While creating a user collector resource, the user can indicate the desired services through the contextualization part of the OCCI creation request, which is passed down to the VM used to run the collector. During its boot, contextualization information is checked and, according to the identified services, proper actions are taken.

UCRE Monitoring and Resource Event Logs The UCRE monitoring service is automatically supported if the user creates a collector and all VMs are created through the portal. All user VMs are preconfigured with the Zabbix agent, which reports data to the collector. The IP address of the collector is passed to each VM through the contextualization service. However, if the user uses the federation API to create VMs, the `usage` element of the `context` tag must be set to `zabbix-agent` and the

collector IP must also be given through the `context` tag as discussed in Sec. 6.1.1.3. If a user is interested in logging all events related to UCRE resources (compute, storage and network), this can be specified while creating the collector resource using contextualization, as indicated in the OCCI request through *log-MQevents-in-zabbix* (see Listing 6.4). To collect all events related to the status of user resources (created, stopped, destroyed, etc.), software implemented in Python responsible for collecting events logs is deployed at the user collector. This software is implemented based on the message queue RabbitMQ[5], where a message queue listener running on the user collector is subscribed to a RabbitMQ server located at federation level. The listener stores all event logs that are made available for the user through the Zabbix GUI and the JSON-RPC API.

Services and Applications Monitoring Users can monitor any service or application deployed on their resources through custom metrics. New measurement metrics can be defined as required to monitor services and applications. Zabbix agents running on those resources are in charge of reporting measurement data of these metrics to the collector. To achieve this, a simple program is implemented that automates the process. It allows the user to add arbitrary metrics in a very simple way. This program is written in Python and can be deployed on any VM. All that is needed is to execute the program, which takes as input the name of the metric and a command used to perform the measurements. This command can be a simple Unix command or even a command to run any tool or software to produce the measures. It automatically reconfigures the Zabbix agent running on the selected VM to start measurement. It also configures the Zabbix server remotely in order to understand the new metrics. User-defined metrics are listed under the hostname of the selected VM along with those metrics supported by default.

Infrastructure Resource Monitoring In order to implement the infrastructure resource monitoring service at each cloud infrastructure, an instance of the Zabbix server is deployed in a VM or a PM, called an *infrastructure collector* (Figure 4.3). The infrastructure collector is in charge of monitoring the whole physical infrastructure. It collects monitoring data from all PMs, and network and storage resources at that infrastructure and from its interconnectivity with other infrastructures. In each physical resource, a Zabbix agent is running that is configured to report data to the respective infrastructure collector. This monitoring setup is used for internal control and management, as well as by federation administrators for assuring the health and performance of the infrastructure. Furthermore, partial information about the physical infrastructures are offered to users through a set of predefined measurement metrics about the physical machines on which users' VMs are running. Read-only

[5] https://www.rabbitmq.com

access to these metrics is offered in the form of a Zabbix template. This is implemented using a permanently running daemon on each user collector that fetches the data from the respective infrastructure collectors through their APIs. The daemon is notified each time the state of any of UCRE's VM is changed (e.g. created, destroyed). This notification is implemented through the subscription of a message queue listener to the message queue server as discussed for the resource event logs. The listener stores each notification in an SQLite database as a new entry in a table called `notifications`. If the notification indicates a creation event of a compute resource (VM), an entry is stored in another table of the database called `computes`. The `compute` table entry includes the name of the created VM, the PM on which it is running, at which infrastructure it is located, and the IP address of the infrastructure collector. If the notification indicates a destroy event of a compute resource (VM), the respective entry is removed from the `computes` table. The daemon periodically (e.g. every 30 seconds) checks the `computes` entries and retrieves information about the PMs from their corresponding infrastructure collectors. If multiple VMs are running on the same PM, information about that PM is retrieved only once. This service is only activated for the user if the *usage* element of the OCCI request to create the collector VM is set with *infra-monitoring-init* (see Listing 6.4). This script will then instantiate the service.

6.1.3. Solution Applicability

This solution is used to monitor a federated cloud environment using homogenous tools across clouds. This, however, limits its direct adoption in a heterogeneous and dynamic federation where different monitoring systems are in place, as mentioned in Sec. 4.1.2. Nevertheless, this solution can still be used as a stand-alone solution for some scenarios. For instance, it can be used to monitor a federation of clouds that is in turn part of a heterogeneous federation but adapts to its common monitoring API. This is the case for one of the experimental use cases addressed in this thesis, where this solution is partially reused in a federation of heterogeneous testbeds (Sec. 7.2).

6.2. Reference Implementation of the Final Architecture

This section presents a reference implementation of the final monitoring architecture, namely MAFIA, whose design is introduced in Sec. 4.3. MAFIA supports various types of monitoring services (Sec. 4.3.1). Their design is discussed in Sec. 4.3.2 in according to the type of consumer. Similarly, the implementation of these services for three main consumers (users, federation administrators and FLS, and federation services) is presented in Sec. 6.2.2. However, the main functional elements of MAFIA are valid for all three types, and are discussed first in Sec. 6.2.1.

6.2.1. Implementation of the Main Functional Elements

To support MAFIA services according to the design principles laid out earlier, mandatory elements have to be provided or implemented by each infrastructure involved in the federation. These includes the deployment of local monitoring tools, the adoption of the common monitoring API and the provisioning of the data following its specification in a common data representation.

6.2.1.1. Local Monitoring Tools

MAFIA is designed in a flexible way to allow infrastructure providers to maintain monitoring tools and solutions already in place, provided that the required services are supported. Each infrastructure can use arbitrary measurement and monitoring tools, as long as they suit its domain, size and types of resources. Indeed, depending on the nature of an infrastructure and the type of monitoring service being provided, some tools might be more appropriate than others. This is out of the scope of this thesis.

To give some idea of the tools used in the implementation, commonly used tools in the experimental use-cases addressed in this thesis include Zabbix, Nagios, collectd, Iperf, the Ping program as active measurement probes, and the Linux monitoring tools `iostat`, `netstat` and `vmstat`.

It's still possible for the federator to recommend a number of powerful tools to be used by those infrastructures that would otherwise lack satisfactory monitoring solutions.

6.2.1.2. Common Monitoring API

The OML framework (discussed in Sec. 2.5.2.3) is used as a reference implementation of the common monitoring API of MAFIA. It defines a flexible protocol called OMSP that is used to describe arbitrary data schemas and to transfer measurement data from distributed clients (measurement and injection points) to one or multiple servers (processing and collection points) as streams according to these schemas.

Upon connection to a server, a client first sends a header followed with measurement data that is serialized using text or binary encoding. The header describes the injection points, along with the schemas of the measurement data. OMSP uses a list data model. The data serialization is done as Delimiter Separated Value (DSV), separated by tab in the text encoding mode, while in binary mode the data is encoded following a specific type of marshaling.

Listing 6.5 shows an example of a valid OMSP header and data streams that are text encoded. This example is used to provide availability status information on a set of components in FUSECO PG.

Listing 6.5: An OMSP header and data streams

```
1  protocol: 4
2  domain: FUSECO
3  start-time: 1441184697
4  sender-id: fuseco.fokus.fraunhofer.de
5  app-name: health_monitoring
6  schema: 0 _experiment_metadata subject:string key:string value:string
7  schema: 1 fiteagle node:string up:int32 last_check:string
8  schema: 2 epc_client node:string up:int32 last_check:string
9  schema: 3 epc_testbed node:string up:int32 last_check:string
10 schema: 4 cloud_testbed node:string up:int32 last_check:string
11 content: text
12
13 10.9578390121 1 0 FITeagle AM 1 2015-09-02T11:05:07.957537+02:00
14 10.9579770565 2 0 EPC Client 0 2015-09-02T11:05:07.957537+02:00
15 10.9580380917 3 0 EPC Testbed 1 2015-09-02T11:05:07.957537+02:00
16 10.9580969811 4 0 Cloud Testbed 1 2015-09-02T11:05:07.957537+02:00
```

Lines 1 to 11 are header information and define meta information about the monitoring data that starts in line 13. In the header, the `domain` tag (line 2) contains the name of the infrastructure from which health monitoring information is provided by a particular client (see `sender-id` in line 4). The client runs a health monitoring application (represented through `app-name` in line 5), which is started at a specific point in time (represented through `start-time` in Unix timestamp). Schema 0 in line 6 is a specific hardcoded stream for metadata. Schemas 1 to 4 are user-defined according to the following definition (see lines 7 to 10 and lines 13 to 16):

`schema: <id> <identifier> <value 1> <value 2> ... <value n>`

Where the placeholders have the following meaning:

- `id`: schema identifier
- `identifier`: unique identifier of the monitored component
- values for example:
  - `node`: defined in a text message to provide the name of the monitored component in a human readable manner
  - `up`: has to be `1` if the component is up, `0` if down, or `2` if the status is unknown
  - `last_check`: indicates the date when the component status was checked

This predefined structure must be kept while pushing the information. It is, however, possible to update any schema when a connection is present or even add new schemas. Schemas must have distinct names as is the case in lines 7 to 10.

Lines 13 to 16 represent the actual monitoring data being pushed, which has the structure defined in the schemas. For example, line 14 belongs to the component

EPC Client with the schema number 2, the resource is down, and the information was generated at `2015-09-02T11:05:07.957537+02:00`.

This example shows a simple scenario as to how OMSP protocol is used. However, the same methodology is valid for all scenarios supported in MAFIA independent of the type of the planned measurements or monitoring services.

Important while using OML/OMSP is to define the MPs or to write suitable wrappers to produce measurement data ready for injection. Defining MPs to produce measurement data is more relevant to users, allowing them to monitor their services or applications. In contrast, the use of wrappers to produce data is most appropriate for supporting most MAFIA services. This is because such wrappers are used by infrastructure providers to gather already produced data from local monitoring tools and provide the data to various consumers. A wrapper fetches measurement data from one or multiple tools through their native APIs (having own data formats) and converts the data into OML streams with the help of supported OML libraries, which then export the data to the identified OML server. As OML includes libraries in multiple languages, such wrappers can be written in any of these languages (as discussed in Sec. 2.5.2.3). Such wrappers are referred to in this thesis as OML wrappers. According to MAFIA architecture (Figure 4.6), a wrapper represents a monitoring adapter or even part of an adapter.

6.2.1.3. Monitoring Adapters

A monitoring adapter compliant with MAFIA specification can be used to either manage a monitoring resource such as a measurement probe (probe instantiation and execution, data processing and delivery, and probe release) or convert data from one interface (format) to another, as discussed in Sec. 4.2. Both cases are implemented for various scenarios as will be discussed later. However, as far as the provisioning of data in a common format is concerned, each adapter used to provide data to its consumer has to deliver it as OML streams. Such adapters can either be responsible for defining and executing MPs and data delivery, or they can act as OML wrappers to wrap data from monitoring tools and deliver it as OML streams.

Wrappers are used to collect data from any data source, e.g. monitoring tools, network traffic statistics, CPU and memory utilization, and input from sensors like temperature. Such generic capability allows writing wrappers for different scenarios. Furthermore, wrappers can be shared and reused by other users or infrastructures as long as they are used to wrap the same set of data from the same tools. Generic, simple examples are provided in Appendix C that can be modified or extended by users for their specific use.

OML supports providing data in a common format following the OMSP specification. However, it does not define specific schemas or vocabulary but rather leaves them to be defined by the user. For instance, the example shown in Listing 6.5 includes

user-defined schemas that contain a set of values and use their own vocabularies (node, up, last_checked). To provide the same service from a different source in a federation, these schemas and vocabulary should be common and shared by all parties involved in order to avoid any misinterpretation and interoperability problems. As this solution will not scale in a large and dynamic federation where MPs, tools and measurement metrics change dynamically, MOFI and the other OMN ontologies are used as a common information model. This model allows common data collection and representation across the federation.

6.2.1.4. Semantic Data Collection and Representation

The use of common, shared definitions of monitoring and measurement concepts and their relations facilities interoperability and allows the data to be provided in a common, meaningful manner across the federation. MOFI describes the common monitoring concepts and their relations in the experimental use-cases addressed in this thesis.

As discussed in Sec. 5.2.1, working towards the goals of sustainability and standardization, MOFI is designed as part of the larger OMN ontology that models federated infrastructures with a focus on the complete experiment life cycle, where MOFI is the OMN-Monitoring subontology. It is, together with the other OMN ontologies, implemented within MAFIA. For reusability and sustainability, the OMN namespace and prefixes are used instead of MOFI for the monitoring onotologies in a fashion similar to all other OMN ontologies. That means instead of using *mofi* as the prefix for concepts and relations defined in the MOFI upper ontology, the prefix *omn-monitoring* is used. Similarly, the prefix *omn-monitoring-data* is used instead of *mofi-data* and *omn-monitoring-metric* is used instead of *mofi-metric*, and so on.

A wide range of infrastructure-, resource- and monitoring-related concepts and relations defined in OMN are implemented within MAFIA, particularly for those components related to data collection and representation, namely OML-related.

A proof-of-concept version[6] of a semantic OML was implemented to support the collection of monitoring data as RDF triples. This version is limited to a specific domain (network measurements) and has not been deployed in any real environment. However, it is used as a basis for the semantic OML implemented in this work. The initial version was dramatically changed and extended in multiple aspects [64], [159]. It was modified to implement the OMN ontology and support the services required within MAFIA. It has also been extended to support semantics in two additional languages (Python and Ruby) besides C, that is natively supported, in order to allow users to use any of the three corresponding client libraries (`liboml2`, `OML4Py` and `OML4R`).

[6] https://github.com/alco90/soml

On the server side, `oml2-server` is extended through additional capabilities for processing and storing OML streams that are collected semantically following RDF-based schemas.

- Upon receipt of a header, the client handler first parses the metadata and forwards the information to a semantic database adapter (called `fuseki-adapter`) for further processing.

- It prepares an INSERT statement for each schema according to the metadata. INSERT statements can be seen as templates used for inserting data, sent following the header, during the whole connection lifetime or at least as long as the schemas are not changed or updated. INSERT statements are prepared in a way so as to allow serialization of the data in RDF triples using the RDF-Turtle data format. That means that, each time a stream is received, the actual monitoring data is inserted into the corresponding INSERT statement that is then ready for insertion serialized as RDF-Turtle.

- The data is then inserted as RDF triples in a semantic triple store. Each insertion (or INSERT statement) contains multiple RDF triples depending on the amount of information included in the corresponding stream.

The Apache Jena Fuseki[7] server is used along with Jena TDB[7] as RDF triple store backend for the semantic OML server. Fuseki server is a SPARQL Protocol And RDF Query Language [220] (SPARQL) endpoint that allows RDF triples to be stored through SPARQL 1.1 Update over HTTP and retrieved through SPARQL 1.1 Query over HTTP.

At the client side, an extension is done in the OML Scaffold program (`oml2-scaffold`) that is natively used to generate the skeleton OML source code for MPs written in C, as discussed in Sec. 2.5.2.3. Scaffold is extended in this work to have a twofold service:

1. To generate part of the OML wrapper code (in C, Python or Ruby) that is used for building the injection of OML streams to follow semantic RDF-based schemas (see line 2 in Listing 6.6) according to a predefined user template (Listing 6.7, which is defined according to OMN monitoring and other related ontologies).

2. To act as a validator to check the correctness of the user-defined semantic schemas against OMN ontologies. The validator gets both as input, user schemas in the form of a template and OMN ontologies, as shown in Figure 6.1.

[7] `http://jena.apache.org/documentation/serving_data/`

Listing 6.6: RDF-based OML streams

```
schema: 0 _experiment_metadata subject:string key:string value:string
schema: 1 used_bandwidth used_bandwidth:double:{omn-monitoring-data:SimpleMeasurement|
    omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:UsedBandwidth}{omn-
    monitoring-metric:UsedBandwidth|omn-monitoring:isMeasurementMetricOf|omn-domain-
    pc:VM}{omn-monitoring-data:SimpleMeasurement|omn-monitoring-
    data:hasMeasurementDataValue|%value%}{omn-monitoring-data:SimpleMeasurement|omn-
    monitoring:hasUnit|omn-monitoring-unit:bitpersecond}{omn-monitoring-
    unit:bitpersecond|omn-monitoring-unit:hasPrefix|omn-monitoring-unit:mega}
    timestamp:datetime:{omn-monitoring-data:SimpleMeasurement|omn-monitoring-
    data:hasTimestamp|%value%} virtualresource:string:{omn-domain-pc:VM|omn:hasURI|%
    value%}{omn-domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}
    physicalresource:string:{omn-domain-pc:PC|omn:hasURI|%value%}
content: text
0.985508918762 1 0 0.002190 2015-10-20 23:22:34+02:00 http://monitoring.service.tu-
    berlin.de/resource/Openstack-1/c277660a-f5c7-4c29-9fdb-590e6e57ecc2 openstack.av.tu-
    berlin.de
```

Listing 6.7: OMN-based OML template

```
app.defMeasurement("used_bandwidth"){ |m|
   m.defMetric('used_bandwidth', :double, 'used␣bandwidth␣of␣monitored␣host',
   [['omn-monitoring-data:SimpleMeasurement','omn-monitoring-data:isMeasurementDataOf','
       omn-monitoring-metric:UsedBandwidth'],
   ['omn-monitoring-metric:UsedBandwidth','omn-monitoring:isMeasurementMetricOf','omn-
       domain-pc:PC'],
   ['omn-monitoring-data:SimpleMeasurement','omn-monitoring-data:hasMeasurementDataValue
       ','%value%'],
   ['omn-monitoring-data:SimpleMeasurement','omn-monitoring:hasUnit','omn-monitoring-
       unit:bitpersecond'],
   ['omn-monitoring-unit:bitpersecond','omn-monitoring-unit:hasPrefix','omn-monitoring-
       unit:mega']])
   m.defMetric('timestamp', :datetime, 'Time␣when␣the␣metric␣is␣measured',
   [['omn-monitoring-data:SimpleMeasurement','omn-monitoring-data:hasTimestamp','%value%
       ']])
   m.defMetric('physicalresource', :string, 'URI␣of␣monitored␣resource',
   [['omn-domain-pc:PC','omn:hasURI','%value%']])
   m.defMetric('virtualresource', :string, 'URI␣of␣virtual␣host␣which␣is␣running␣on␣the␣
       monitored␣physical␣host',
   [['omn-domain-pc:VM','omn:hasURI','%value%'],
   ['omn-domain-pc:VM','omn-lifecycle:childOf','omn-domain-pc:PC']])}
```

Figure 6.1 shows an overview of the implementation of the semantic OML delivered by the work done in this thesis together with a set of examples of monitoring and measurement tools (e.g. Zabbix, Ping, Traceroute, Ierf) and capabilities (e.g. OpenStack API, Ceilometer) used to produce raw measurement data. OML wrappers that are part of OML clients together with the proper OML libraries are used to gather monitoring data from these tools and convert them into OML streams and send them to the server. The server stores the data in Jena Triple Store through SPARQL updates over HTTP. However, in order for these wrappers to export the data semantically following RDF based schemas, they need to use OMN models. For this to be achieved, a user has to define RDF-based schemas through templates that

includes the required measurement metrics and the associated monitored resources. An example of such a template (written in Ruby) is shown in Listing 6.7.

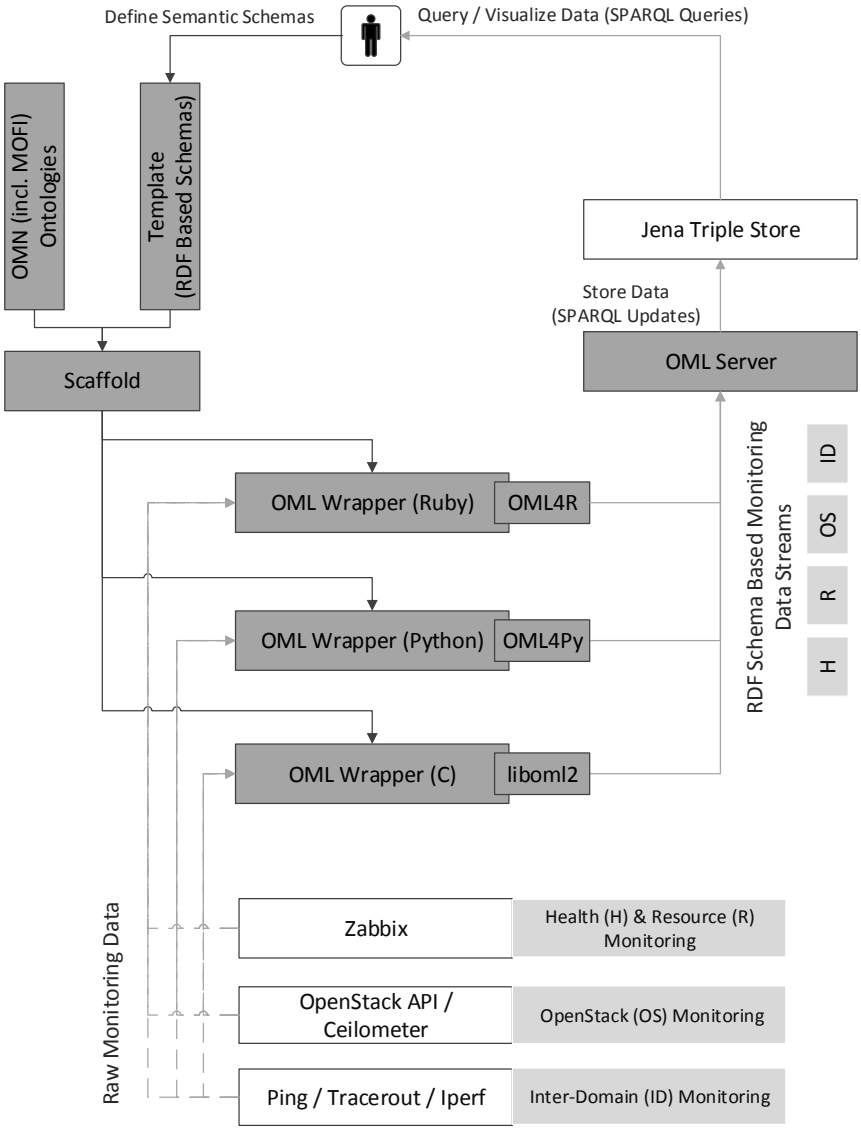


Fig. 6.1.: Semantic OML implementation

In order to simplify the scenario in Figure 6.1, it's assumed that the user who defines the template is the same as the one who will then query the data. However in practical use, these can either be the same user or two different users. In the former case, a user uses OML for performing measurement related to their services and applications and then utilizes the data. As an example of the latter case, an infrastructure provider defines a template that is used to export data about infrastructure resources; an authorized user can query the data based on resource IDs (limited to the given IDs) or resource types (gets data about all resources of the given type).

The adoption of semantic OML together with the use of OMN would not need much manual change if a classic OML is already in place. At the server side, an OML server can be upgraded to the new semantic version that supports RDF serialized data streams. Furthermore, there is no need for any manual change to go back to the use of a classic OML, as the new version can handle both schemas (semantic and traditional). At the client side, it's not necessary to change or upgrade any of the traditional OML client libraries in place. An exception is in the latest version of OML4Py `(v2.10.4)`. This version checks the format of the used schemas in the traditional way. It is extended to handle both schemas (semantic and traditional). There is no real overhead here as this library is just a Python script that can be replaced. However in general, all that has to be done at the client side is to write MPs or OML wrappers in to export streams following predefined RDF-based schemas. This process is even facilitated via the use of `oml2-scaffold`.

6.2.1.5. Data Access and Visualization

Users can query or visualize their data in multiple ways. If the classic OML framework is used that stores data in either an SQLite or PostgreSQL database, the user can use any SQL client to query the data. Furthermore, several visualization tools can be used on top of these databases. In the case of PostgreSQL, for instance, a range of visualization and GUI-based tools[8] are available. As MAFIA supports different kinds of users with various interests, the decision of how to access the data is left to the user.

In the case of the semantic OML, data is retrieved through the use of SPARQL 1.1 Query Language. Fuseki's SPARQL Query engine supports by default exporting semantic triples in the form of tables or in other formats such as XML, JSON and CSV. It's hard for a human user to understand the data as it is stored as triples and identified through unique and specific URIs. To provide a data visualization capability, Sgvizler[9] is used. It coverts the RDF triples queried from the *Jena TDB* into graphs and charts. Sgvizler is a Javascript file (`sqvizler.js`) that can be placed in the directory where Fuseki is deployed under the `pages` folder. All ontology prefixes used in the semantic OML (MOFI, OMN, W3C Time, and others) are written in this file to avoid using them in each SPARQL query. Finally, this file can be either embedded in a portal or used in any Hyper Text Markup Language (HTML) page. The latter is used in the this implementation where the new created page is added to `fuseki.html`. By doing so, a new link to the visualization appears in the Fuseki GUI.

[8] https://wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools
[9] https://www.w3.org/2001/sw/wiki/Sgvizler

6.2.2. Implementation of Monitoring Services for Various Consumers

In Sec. 6.2.1, the implementation of the main functional elements of MAFIA was discussed. This section presents the implementation of MAFIA services and how they are made available for their consumers.

As all services have to provide monitoring data via OML across the federation, it is assumed that each infrastructure has deployed suitable monitoring tools in charge of data acquisition and is able to forward the data as OML streams.

MAFIA is designed in an extensible manner so that arbitrary monitoring services can be deployed for various consumers [143]. However, a set of services are designed in this thesis that are consumed by different consumers, grouped into three categories (see Sec. 4.3.2). Their implementation is discussed in the following three sections.

6.2.2.1. MAFIA Services for Users

Monitoring services for users can be implemented in two different approaches: generic ad user-friendly (Sec. 4.3.2.1) [143]. The user-friendly approach is the reference implementation delivered by this work. However, a proof-of-concept implementation that demonstrates the feasibility of the generic approach is carried out and is discussed briefly below.

Generic Implementation:

This approach assumes that an infrastructure offers monitoring resources for users like other classic resources (e.g. VM, sensor, WiFi Access Point) that can be requested on-demand. Examples of monitoring resources include measurement probes, data converter, data exporter, data collector, and data viewer (API, GUI). In this implementation, in contrast to the user-friendly implementation, all monitoring and measurement setup and configuration are performed by the user.

This generic approach was implemented using the Teagle framework, which was used for registering and provisioning monitoring resources [94]. Teagle was selected due to its generality and resource agnosticism, i.e. anything can be a resource that can be managed, like software, a server or even a platform. It includes multiple central services for resource description, registration and provisioning across federated domains. Infrastructures advertise resources described with their functionalities. While creating a UCRE the user requests suitable resources that will be used to perform measurements and collect data. User requests are dispatched to the respective infrastructures. Monitoring resources are then provisioned along with classic resources to form the UCRE. In Teagle, each of the offered resources is managed via a resource adapter.

In the proof-of-concept implementation, the Packet Tracking tool is used, which is a passive measurement solution for network path and quality monitoring. It is used for performing traffic monitoring between nodes deployed in a multi-domain environment. It includes three main components: `impd4e` (passive measurement probe), `matcher` (data processing and collection) and `netview` (data visualization). Two resource adapters were developed within Teagle: one for the `impd4e` and the other one for the `matcher`. Both resource types are offered through Teagle along with other resource types, such as VM, server, router and database. `Netview` as an executable Java Archive (JAR) file can be deployed anywhere (e.g. on the matcher machine) to get the data from the `matcher` to be visualized.

Teagle includes the Virtual Customer Testbed Tool [221] (VCTTool), through which a user can setup a UCRE. During setup, the user can identify the type and number of the required resources (e.g. VMs, routers), in which infrastructures they should be deployed, and their configuration and dependencies. The user can request an instance of the `impd4e` to be deployed on each of the VMs between which delay and network path quality measurements are to be performed. It's mandatory to deploy an instance of the `matcher` in advance as its IP is required by all probes to report data. Teagle includes an orchestration engine that supports the provisioning of resources following a pre-defined order to ensure dependencies.

User-friendly Implementation:

A user-friendly implementation enables automatic setup of monitoring services. This allows users to get monitoring data about their UCREs and their accompanying infrastructure resources on an on-request basis without any need of the users' involvement in setup and configuration (see Sec. 4.3.2.1) [143].

This implementation requires some effort at the infrastructure level. First of all, it assumes that the infrastructure already has some means to actually measure infrastructure-related metrics. Information about infrastructure resources is offered to users as a service in addition to information about their UCRE resources. Offering these services is achieved analog to the way classic resources are being offered.

In the FI experimentation field that is addressed in this thesis as an experimental use case, the SFA Aggregate Manager (AM) and Clearinghouse (CH) APIs are used as a de-facto standard for resource discovery, provisioning and termination across federated infrastructures. SFA is accompanied by the GENI Resource Specification[10] (RSpec) that is used for describing resources in a common manner.

FITeagle, one of the SFA implementations, is used in this work. Using FITeagle as an infrastructure manager, classic resources offered by the concerned infrastructure are described via RSpec advertisements. GENI RSpecs are XML-based documents

[10] `http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs`

and one RSpec is provided per resource type. In FITeagle, each resource type is described via an RSpec and managed by a resource manager.

FITeagle was extended within this thesis to implement some MAFIA functionalities, namely the advertisement and provisioning of monitoring capabilities associated with the classic resources offered. These are illustrated in Figure 6.2 through the steps 1 to 4.

An example of an RSpec advertisement for the resource type VM as managed by FITeagle is shown in Listing 6.8. The capability of offering monitoring services is indicated through offering *MonitoringService* in the `monitoring` element. This means that monitoring data related to the described resource (OpenStack VM in this case) can be provided upon request as OML streams to a given collection endpoint.

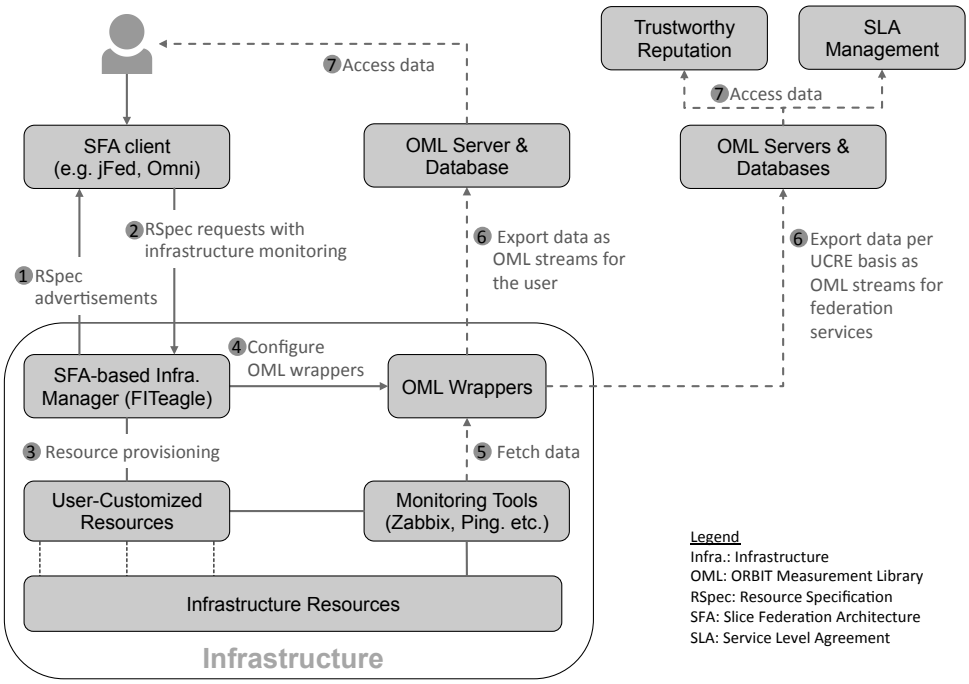


Fig. 6.2.: Implementation of MAFIA services

Using any SFA client, such as jFed[11] (jFed) or the GENI command line tool Omni[12] (Omni), a user can discover resources offered by any infrastructure participating in the federation. A user can on-demand request resources, and optionally identify interest in having monitoring services if offered (step 2 in Figure 6.2). Listing 6.9 shows an example of an RSpec request to create an instance of a VM with monitoring services. As the data has to be collected across the federation as OML streams, the user is obliged to identify the collection endpoint (i.e. URI) that is capable of receiving and understanding OML streams.

[11] http://jfed.iminds.be
[12] http://trac.gpolab.bbn.com/gcf/wiki/Omni

Listing 6.8: RSpec advertisement of VM with monitoring capabilities

```
<rspec
   type="advertisement"
   generated="2015-10-06T11:40:46.078Z"
   expires="2015-10-06T11:40:46.078Z"
   xmlns="http://www.geni.net/resources/rspec/3">
   <node
      component_id="urn:publicid:IDN+monitoring.service.tu-berlin.de+node+http%3A%2F%2
          Fmonitoring.service.tu-berlin.de%2Fresource%2FOpenstack-1"
      component_manager_id="urn:publicid:IDN+localhost+authority+cm"
      component_name="Openstack-1"
      exclusive="false">
      <sliver_type name="http://monitoring.service.tu-berlin.de/resource/Ubuntu-64bit"/>
      <sliver_type name="http://monitoring.service.tu-berlin.de/resource/2048MB_20GB"/>
      <sliver_type name="http://monitoring.service.tu-berlin.de/resource/1024MB_10GB"/>
      <sliver_type name="http://monitoring.service.tu-berlin.de/resource/512MB_5GB"/>
      <location latitude="52.516377" longitude="13.323732"/>
      <ns2:monitoring type="http://open-multinet.info/ontology/omn-monitoring#
          MonitoringService"/>
    </node>
</rspec>
```

Listing 6.9: RSpec request to create a VM with monitoring services

```
<rspec
   type="request"
   generated="2014-07-11T10:20:39Z"
   xsi:schemaLocation="http://www.geni.net/resources/rspec/3␣http://www.geni.net/
       resources/rspec/3/request.xsd"
   xmlns:client="http://www.protogeni.net/resources/rspec/ext/client/1">
   <node
       client_id="monitoringTest_VM"
       component_id="urn:publicid:IDN+monitoring.service.tu-berlin.de+node+http%3A%2F%2
           Fmonitoring.service.tu-berlin.de%2Fresource%2FOpenstack-1+Ubuntu-64bit"
       component_manager_id="urn:publicid:IDN+monitoring.service.tu-berlin.de+authority+
           am"
       component_name="Openstack-1"
       exclusive="false">
       <sliver_type name="http://monitoring.service.tu-berlin.de/resource/Ubuntu-64bit"/>
       <ns2:monitoring type="http://open-multinet.info/ontology/omn-monitoring#
           MonitoringService" uri="http://130.149.22.139:3003"/>
   </node>
</rspec>
```

This resource has to be created by the user in advance as part of the UCRE before creating the planned classic resources. Upon receipt of the user request by FITeagle, two actions are taken.

First, the respective resource adapter processes the request. In this example, an OpenStack resource adapter is in charge of processing the VM resource. It has to perform resource provisioning (step 3 in Figure 6.2). It interacts with the OpenStack API and creates a VM. The results are then pushed back to the FITeagle message bus. Information, such as the given VM ID, its current status (in the the case of creation, *started*), and the URI of the identified user OML collection resource, is used

by the monitoring integrated service that is implemented as part of this thesis within FITeagle to support monitoring services. Such information is used by this service for two purposes:

- Using the VM ID, to contact the OpenStack API to retrieve the ID of the PM that hosts the created VM.

- If monitoring services are requested, to instantiate and configure a suitable OML wrapper to provide monitoring data to the user (step 4 in Figure 6.2). This wrapper periodically (on a regular basis, e.g. every 30 seconds) fetches monitoring data already collected by local monitoring tools (Zabbix is used) about both machines using their IDs (step 5 in Figure 6.2) and exports the data as OML streams to the user collection resource using the given URI (step 6 in Figure 6.2) during the UCRE lifetime.

The collection resource can be an OML server together with a database backend (Jena[7] Toolkit in the case of semantic collection). This resource can be optionally offered by the infrastructure (or any other infrastructure participating in the federation) as a resource in the same way as any other classic resource. This resource can be deployed in a form of a VM in cloud infrastructures. The user could use such a collector as a collection endpoint for monitoring data collected from two different sources: monitoring information about the infrastructure resources received from infrastructures, and data generated by measurements related to the user's services and applications.

The user can finally access data (step 7 in Figure 6.2) using tools such as SPARQL Query or visualize the data as discussed in Sec. 6.2.1.4. As long as the collection resource is owned by the user for exclusive use, there are no access control and privacy issues.

Once the VM resource is deleted or its lifetime is over, a notification message is provided by the OpenStack resource adapter through the FITeagle message bus. This message includes information like the VM ID and its current status (*deleted*). It is used by the FITeagle monitoring integrated service to re-configure the OML wrapper to stop pushing data to the user.

There are no special implementation efforts required by users. All that a user needs to do is to ensure the existing of an OML collection resource to receive data. It has to be provisioned prior to requesting UCRE resources.

At the federation level, some effort is required during implementation. Taking the Fed4FIRE federation as an example, user and UCRE (referred to as a *Slice* in SFA) authentication and authorization are based on X.509 v3 certificates that are issued by the member (user) and slice authorities respectively. Furthermore, a Web portal can be provided to allow users to interact with all these resources in a more user-friendly

manner, from resource discovery and provisioning up to data collection and access. In the Fed4FIRE federation, MySlice is used to power the Fed4FIRE Web portal.

6.2.2.2. MAFIA Services for Federation Administrators and FLS Monitoring Dashboard

As discussed in Sec. 4.3.2.2 [143], health and status information about federated infrastructures are of major importance for the federation administrator in order to ensure availability of infrastructures and their main services. This information is also required by the federation's FLS system, which displays the information through its monitoring dashboard.

Providing such information does not require significant effort at infrastructure level, as monitoring tools are probably already deployed to monitor infrastructure resources and services for internal administrative demands. If it is not the case, it's left to each infrastructure to deploy suitable tools depending on its type, size and the types of resources offered. Only information that reports the status of the key components in each infrastructure is required.

At each infrastructure, such information is obtained through measuring the availability status of each of the key components. The data is then reported via an OML wrapper that is permanently running. The wrapper periodically (e.g. every 5 minutes) fetches the required data from local tools and exports it as OML streams to a central OML collection resource located at the federation level. The data is then accessed by the federation administrators and the FLS system. The FLS system calculates the overall status of each infrastructure and displays this information through its monitoring dashboard.

6.2.2.3. MAFIA Services for Federation Services

A number of federation services require monitoring information as part of their functionalities. They have different requirements for monitoring information (Sec. 4.3.2.3). For each service, the types and frequency of the measured metrics are identified and arranged between the individual infrastructure providers and the service developer [143]. Various sets of metrics are measured, and thus different kinds of information is expected from one infrastructure to another due to their heterogeneity. Federation services receive information exported as OML streams from individual infrastructures provided that each service has at least one OML collection resource. This depends on the deployment setup of each service, whether distributed or central (Sec. 4.3.2.3).

The implementation of these services itself is out of scope of this thesis, however, because they are served with monitoring information, they will be discussed briefly.

For instance, in the case of the SLA management service, upon receipt of a request to create UCRE resources, FITeagle deploys a specific OML wrapper. It periodically

(e.g. every 5 minutes) fetches monitoring data of predefined metrics per UCRE basis only during its lifetime (step 5 Figure 6.2) and exports this data to the OML collection resource already running for the SLA management service (step 6 in the right side of Figure 6.2).

The same procedure is also valid for the trustworthy reputation service. The only difference is the metrics measured and the frequency of the delivered data. Of course another OML wrapper is used.

Other federation services, such as reservation or brokerage, require historical information for resource characteristics. The delivery of such information is not associated with any UCRE or user. Therefore, the data can be provided in various ways, either through RSpecs or exported as OML streams. In the latter case, an OML wrapper is used that is permanently running and periodically reports data. To reduce the overhead and the amount of data, the frequency of data delivery can be as large as possible, e.g. every hour.

6.3. Summary

This chapter presented a reference implementation of MAFIA along with the MOFI information model in Sec. 6.2. The implementation of the initial architecture design was presented first in Sec. 6.1, as this can be deployed as a stand-alone solution for any cloud environment or cloud federation that has no monitoring solution in place. Furthermore, an environment using the implementation of the initial architecture design can also be part of a heterogeneous federation that adopts the final reference implementation of MAFIA, discussed in Sec. 6.2, but use OML and MOFI model on top of that to export monitoring data in a common, meaningful way to the outside world.

This chapter focused on the final and reference MAFIA implementation. The implementation of the main architectural functional elements were discussed (Sec. 6.2.1), these are valid for all services provided for multiple consumers. Thereafter in Sec. 6.2.2, the implementation of MAFIA services was discussed according to consumer type.

The applicability and effectiveness of these implementations are discussed in Chapter 7.

7 Validation and Evaluation

THIS chapter presents the validation and evaluation of the solutions delivered by this thesis. Multiple design evaluation methods are available. Hevner et al. categorize the well known methods as observational, experimental, analytical, testing and descriptive [160]. Some methods from the first three categories are selected according to the design requirements, as well as selected evaluation metrics. These metrics are as follow:

- The *field study* as one of the observational methods defined in [160] is selected, and is used to monitor the use of the designed solutions in multiple projects.

- Amongst the two experimental methods defined in [160], namely *simulation* and *controlled experiment*, the latter is selected in this work to study the usability and effectiveness of the designed solutions.

- Some analytical evaluation methods are chosen to study the *effectiveness*, *quality and correctness*, and *performance* of the implementation (presented in Chapter 6) of the designed monitoring architecture (MAFIA, introduced in Chapter 4) and the associated information models (MOFI, presented in Chapter 5).

In this chapter, those requirements identified in Chapter 4 are validated. In addition to this, a comparison between the solution delivered by this thesis with the state-of-the-art solutions is given.

7.1. Observational Evaluation

The *field study* is chosen as an observational evaluation method as the solutions and methods delivered by this thesis are used in multiple research projects. They have thus evolved in iterations according to their demonstration, impact and feedback from these projects. Some concepts and implementations are even reused or extended in different projects. A high level overview of these projects along with the use and impact of the solutions delivered by this thesis are discussed in the following subsections.

7.1.1. The FP7 ICT BonFIRE Project

The BonFIRE project offers a federated cloud testbed supporting large-scale experimentation of applications, services and systems over multiple, geographically distributed and heterogeneous cloud and network testbeds.

At the core of BonFIRE are seven geographically distributed cloud testbeds: (EPCC (UK), INRIA (France), HLRS (Germany), iMinds (Belgium), HP (UK), PSNC (Poland) and Wellness Telecom (Spain)). Together these offer around 660 computing cores with 1.5TB of RAM and 34TB of storage. An additional 2300 cores can be added to BonFIRE on-request [64], [80], [161].

These testbeds are heterogeneous in terms of cloud managers (OpenNebula[1][162], HP Cells[163] and VMWare[2]), hypervisors (Xen, KVM and ESX) and the types of hardware. Besides cloud resources, BonFIRE allows access to the Virtual Wall

[1] http://opennebula.org
[2] http://www.vmware.com

emulated network facility[3], as well as the FEDERICA[164] and the AutoBAHN Bandwidth on Demand[4] service of GÉANT[5].

Testbed infrastructures are interconnected through the projects's own network, called BonFIRE WAN and, thus, any VMs in any testbeds can connect to each other.

This network is used by BonFIRE services to manage and monitor VMs. Each testbed is interconnected to every other testbed via a direct VPN connection.

Amongst the federation models discussed in Sec. 2.4.1, the BonFIRE federation architecture can be seen as a mix of a central management system and a heterogeneous federation. BonFIRE uses a central Resource Manager to expose a homogeneous API and interact with the management systems of the individual federated testbeds with the help of an Enactor.

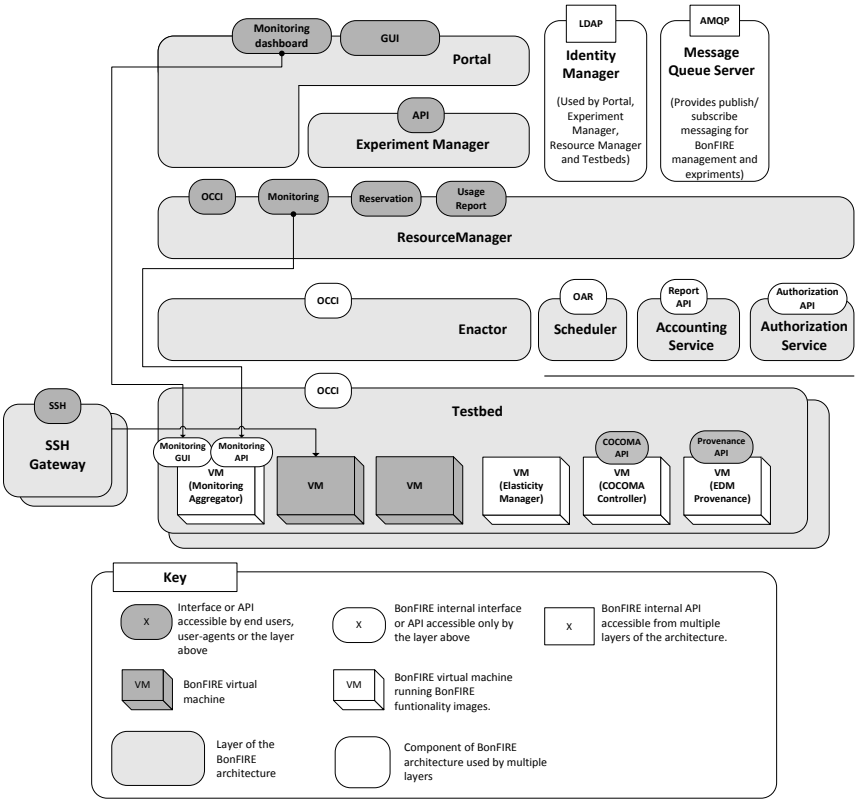


Fig. 7.1.: BonFIRE federation architecture [161]

Figure 7.1 illustrates the main components of the BonFIRE federation architecture. The Resource Manager exposes an OCCI interface for remote management interactions with the infrastructures. It is the lowest level interface provided for experimenters to give them full programmatic control over their experiments, similar to those

[3] https://www.ibcn.intec.ugent.be/content/ilabt-virtual-wall
[4] http://geant3.archive.geant.net/service/autobahn/Pages/home.aspx
[5] http://www.geant.net/Pages/default.aspx

offered by IaaS commercial cloud solutions. The OCCI acts as the user API for the whole experiment lifecycle management (create, stop, resume, shutdown, and delete resources). The Enactor acts as a mediator that receives OCCI requests from the Resource Manager and transforms them into a suitable format for the appropriate testbeds through adaptors. BonFIRE also provides a higher level interface through the Experiment Manager that allows an experiment as a whole to be described. An experimenter can specify the initial deployment of the experiment's resources in a single document called the experiment descriptor using OVF or JSON format. The Experiment Manager takes this description and manages its execution by performing multiple calls to the Resource Manager. Experimenters can also use the Web portal to create and manage their experiments in a user-friendly manner.

BonFIRE aimed to allow experimenters to control and monitor the execution of their experiments to a degree that is not found in commercial clouds [17], [64], [80]. Concerning monitoring support, the initial monitoring architecture design and its implementation discussed in Sec. 4.1.1.3 and Sec. 6.1 respectively have been adopted by BonFIRE [64]. Thus, the BonFIRE monitoring system supports aggregating a multitude of measurements from resources of different administrative clouds in a unified manner, monitoring resources from heterogeneous domains on both the network and infrastructure level, and providing support that is able to operate across large numbers of end-to-end resources at both the service and the application levels.

The effectiveness, utility and performance of this system have been evaluated and discussed in different dimensions. For instance, it has been used for supporting the elasticity-as-a-service provided by BonFIRE to auto-scale cloud resources up and down [139]. Furthermore, significant experiments that were conducted in BonFIRE benefitted from the monitoring services offered, e.g. [165] and [166].

To shed light on the system utility the extent of performance and the benefit of infrastructure monitoring for experimenters, an experiment was conducted [64] and is discussed as follows. The creation of an experiment along with monitoring services was achieved through sending a number of individual OCCI requests to the BonFIRE Resource Manager's API. These requests first created the actual experiment and subsequently deployed and configured a VM to serve as the user monitoring collector (referred to in BonFIRE as Monitoring Aggregator). This experiment was created through the BonFIRE Web Portal and the monitoring service was enabled by choosing which BonFIRE cloud the aggregator needed to be deployed in. Furthermore, the infrastructure monitoring service was also enabled, as BonFIRE allows experimenters to enable or disable any monitoring service on-demand through the Portal.

So far only the experiment container (i.e. UCRE) and the aggregator VM were created and BonFIRE Portal provided the experimenter seamless access to the aggregator's GUI through the BonFIRE Portal. The VMs required to run the experiment were then created and were automatically monitored along with their hosting PMs.

BonFIRE offers reservation of PMs, called clusters, for exclusive use. In this experiment, a cluster was reserved to ensure that only experimenter VMs were running on it. On this cluster, three VMs were created. At this point, the experiment comprised one aggregator VM running on a classic PM and three VMs running on the reserved cluster. On the three VMs, benchmarking was executed, not for testing performance but only to investigate and show the behavior of performance metrics like CPU load within the VMs and the cluster. The IOzone[6] benchmark was used for this purpose, although it is in practice usually used for testing file I/O performance on various file systems. Still, the operations performed by the benchmark consume processing power and the CPU load will vary based on the number of operations being executed by the benchmark and the file system that is employed. The benchmark was run for more than two days. Figure 7.2 shows the behavior of the CPU load within the physical cluster represented by the black line, and the average of the CPU loads within the three VMs represented by the blue line. The result shown in the first graph is for one hour; in the second graph, for twelve hours. As expected, loads are almost similar, which indicates that experimenters can get meaningful real-time information about the underlying infrastructure that can be used as support for suitable decision-making. Additionally, from the results, it is clear that the CPU processing consumed by monitoring Zabbix agents is almost negligible.

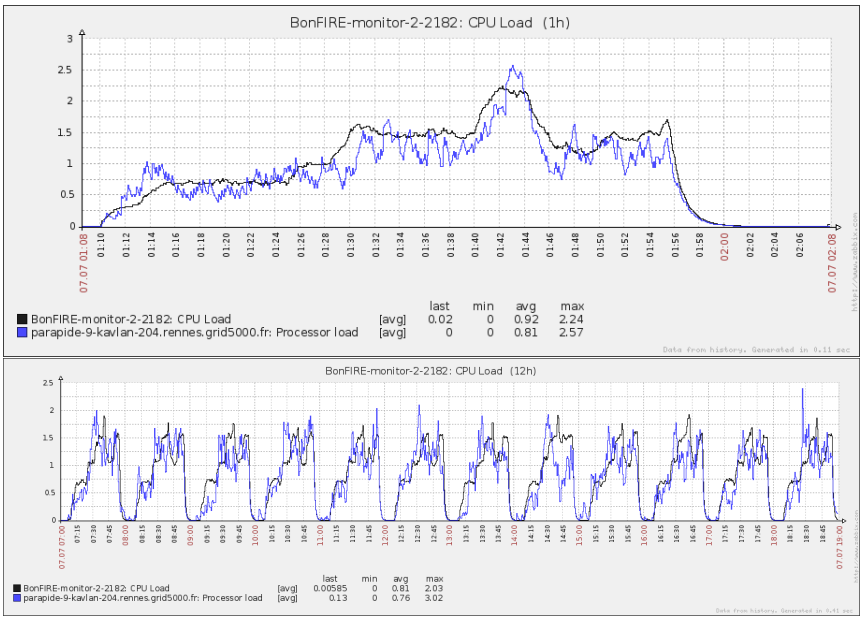


Fig. 7.2.: CPU load within a physical machine (blue) and the average CPU load within three virtual machines running on the physical machine (black) [64]

[6] http://www.iozone.org

7.1.2. The FP7 ICT FI-STAR Project

The implementation of the initial monitoring architecture is partially re-used and extended within the FI-STAR project. This is delivered as the Monitoring Service SE[7], which is one of the FI-STAR platform components. As mentioned in Sec. 2.3.3, beside a set of FIWARE GEs, the FI-STAR platform consists of a number of components referred to as SEs. The Monitoring Service SE is in charge of monitoring the FI-STAR platform components (GEs and SEs), which are deployed as VMs in one or multiple clouds, and the applications running on the platform.

The objective of the FI-STAR project is to facilitate and accelerate the development of healthcare applications by providing a number of GEs and SEs that can be composed on-demand by application developers. This allows develop and build their own platforms with as quickly as possible and to create code to deliver their services for hospitals and patients. During the project lifetime, seven early trials are deployed and executed across Europe, serving more than four million people.

From this perspective, application developers deploy various instances of FI-STAR platform depending on their needs. As OpenStack-based cloud infrastructures are used, FI-STAR platform instances are deployed as isolated tenants. Each instance has a distinct set of GEs and SEs.

The Monitoring Service SE provides monitoring information for two main consumers: application developers and cloud administrators. Developers can monitor the availability, health and performance (e.g. CPU, disk and memory consumption) of the used GEs and SEs, as well as application-related metrics that are supported through customizable services, which allow developers to define own metrics. If a developer is interested in monitoring services, they can deploy the Monitoring Service SE part of their FI-STAR platform instances. Cloud administrators can deploy an instance of the Monitoring Service SE to monitor the usage of each FI-STAR platform instance, e.g. the number of tenants, the number and type of the used GEs and SEs in each tenant, number of used CPUs, used memory in each tenant, and more.

Two extensions were written in order to support FI-STAR need, in addition to the implementation used in BonFIRE. First, a set of probes have been developed to deliver monitoring information from the virtualization layer (OpenStack), that is located between the physical infrastructure and the cloud resources. Second, in addition to the JSON-RPC API and the Web frontend supported by Zabbix, a REST API is implemented within the context of the FI-STAR project, allowing users to get the data over HTTP.

[7] http://catalogue.fi-star.eu/enablers/monitoring-service

7.1.3. The FP7 ICT OpenLab Project

The final monitoring architecture delivered by this thesis, in particular the generic implementation of MAFIA for providing monitoring services for users, has been initially deployed and validated within the OpenLab project.

As mentioned in Sec. 2.3.2, OpenLab focused on the design and development of a federation framework operating across heterogeneous domains. Within the OpenLab project as discussed in Sec. 6.2.2.1 and [94], a proof of concept implementation of the generic approach of MAFIA has been validated and demonstrated. Figure 7.3 illustrates the workflow of a demo scenario, where an experimenter uses the Teagle Virtual Customer Testbed (VCT) tool to setup an experiment as shown in Figure 7.4 [167] (the demo video[8] is publicaly available).

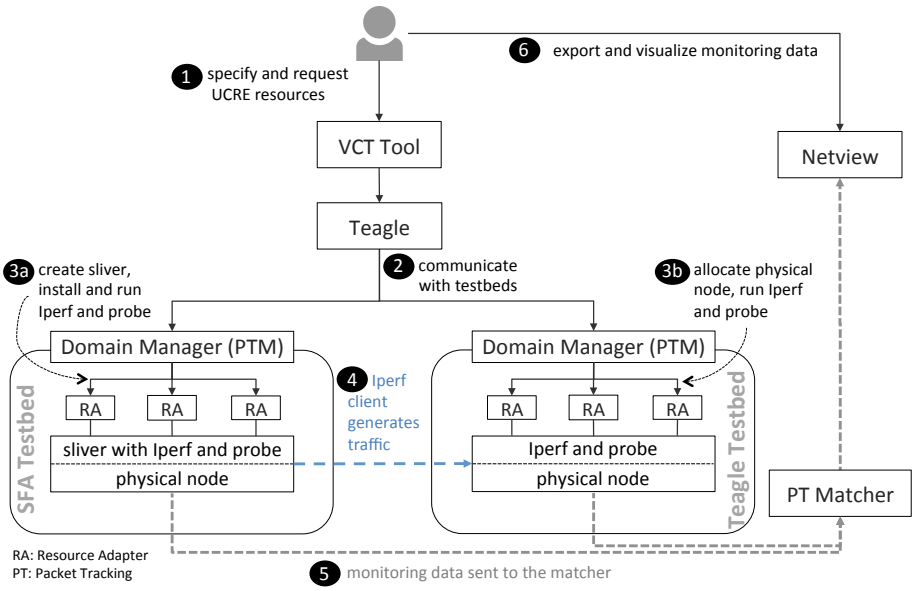


Fig. 7.3.: Workflow of an experiment setup (using Teagle) and execution

The Packet Tracking monitoring tool, Iperf tools and two physical nodes are used. They are offered by two testbeds (one is managed by the Teagle framework and the other is managed by the SFA framework to which Teagle is interfaced). To demonstrate the flexibility of the generic implementation approach of MAFIA, two configurations were tested in this scenario. First, the physical node offered by the Teagle-based testbed was already provisioned. It was then used by the experimenter, on which the preinstalled Packet Tracking probe and Iperf server were run. Second, on the node offered by the SFA-based testbed, a VM (sliver in SFA terminology) was created and a Packet Tracking probe and an Iperf client were downloaded from the Internet, installed and run. For managing these resources, four resource adapters have

[8] http://www.av.tu-berlin.de/menue/research_development/tools/packet_tracking/

been implemented for probe, Iperf, physical node, and SFA sliver. As Teagle works in an orchestrated way, probes and Iperf were run after nodes. This is managed at the VCT level via dependency relations as represented in Figure 7.4 via black connections.

Upon having all resources deployed, the Iperf client connected and sent traffic to the Iperf server running on the other node to generate artificial traffic between the provisioned nodes. Packet Tracking probes reported monitoring data to the Packet Tracking matcher, which was in this scenario up und running prior to the experiment. The experimenter locally deployed Netview (Packet Tracking's visaulization tool) to track and visualize packet traffic and delay statistics as shown in Figure 7.5.

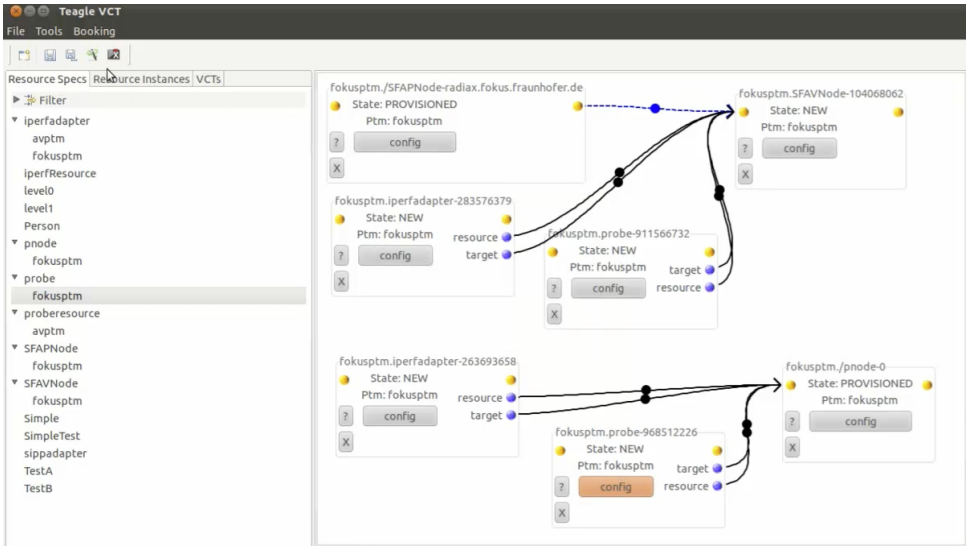


Fig. 7.4.: Resource allocation through Teagle's VCT

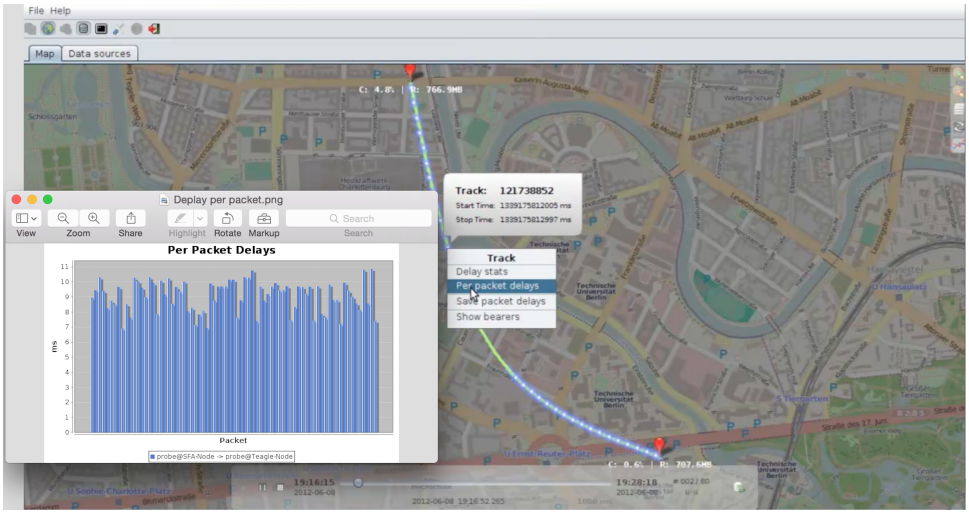


Fig. 7.5.: Packet traffic and delay visualization between two nodes through Netview

Packet Tracking is integrated with the Open Evolved Packet Core[9] [202] (OpenEPC) [168], [169], allowing users to track and study the performance of OpenEPC components and chose suitable configurations and setup to meet the concerned QoS requirements. Both OpenEPC and Packet Tracking are also made available via Teagle, allowing experimenters to provision them on-demand. Packet Tracking probes are deployed on each of the OpenEPC components.

7.1.4. The FP7 ICT XIFI Project

The initial results of this thesis have contributed to the federation architecture design and sustainability plans of FIWARE Lab [70], in particular in terms of monitoring system, which is one of the main systems delivered by the XIFI project.

Among the federation models discussed in Sec. 2.4.1, FIWARE Lab has adopted a hybrid federation model covering aspects of the one-stop-shop model and the integrator model [70]. At the time writing, the FIWARE Lab federation comprised over 17 cloud-based infrastructures distributed across Europe. Based on a deployment configuration, two types of infrastructures are to be distinguished: master and slave.

Figure 7.6 illustrates the FIWARE Lab federation architecture. The lower part contains the components deployed on each infrastructure (both master and slave), and the upper part contains the components deployed only on the master infrastructures. A master infrastructure is the one where, in addition to the features deployed on slave infrastructures, the centralized parts of the federation services (i.e. the components needed to manage the federation) are deployed. A slave infrastructure is the one where, only the software needed for deploying and managing user resources and services, is installed. This software comprises the cloud computing (OpenStack-based), monitoring and security functionalities.

For monitoring the individual deployed VMs (either those hosting GE instances or traditional ones) across the federation, the FIWARE Lab monitoring system is designed following the same methodology designed in the initial monitoring architecture design delivered by this thesis, in terms of the monitoring setup and deployment. Each VM is deployed with a preinstalled monitoring probe (agent) that reports monitoring data to a monitoring collector. One monitoring collector instance is deployed in each cloud infrastructure that collects monitoring data pushed from all probes. An auto-registration service is enabled so that each probe is then automatically registered to the collector, requiring no intervention from the user [19], [149].

However, the major difference from the initial monitoring architecture design delivered by this thesis, which is identified as an architectural limitation based on practical experience (see Sec. 4.1.1.4), is that each FIWARE Lab cloud infrastructure provider can use any monitoring tool to provide the data. This means that there

[9] http://www.openepc.net

us no need to use homogenous tools to provide the data. However, monitoring data has to be collected in a common format across the federation. Therefore, the Open Mobile Alliance (OMA) Next-Generation Service Interface [194] (NGSI) is used in FIWARE Lab for this purpose. This facilitates having an easy to use and sustainable federation architecture [70]. It makes it possible to integrate any cloud infrastructure with its own monitoring system(s) in place. This infrastructure will need to adopt a specific set of adapters to become part of the federation. A set of adapters are implemented specific to a set of application areas (e.g. virtual and physical machines, and cross-domain inter-connectivity). These adapters are accompanied by a common data model to provide the data in one format [19], [149].

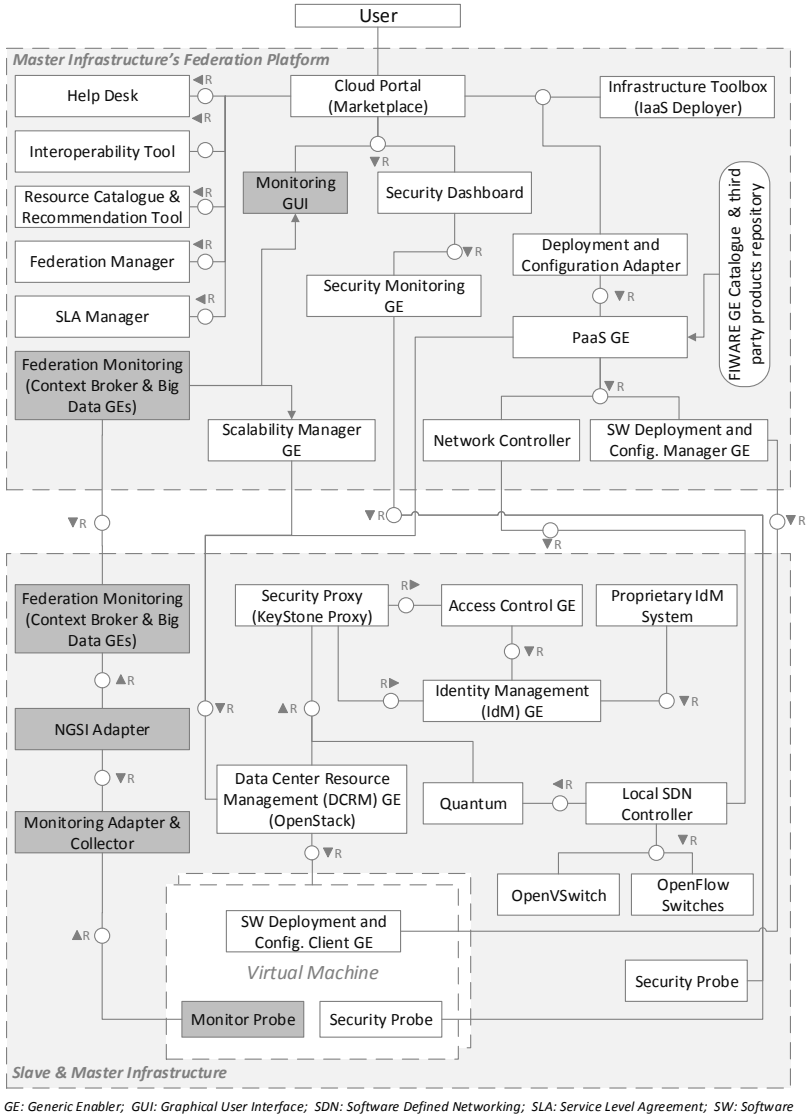


Fig. 7.6.: FIWARE Lab federation architecture [70]

7.1.5. The FP7 ICT Fed4FIRE Project

The Fed4FIRE project aims to federate all FIRE testbeds to build a large-scale, widely distributed, heterogeneous and powerful facility for FI experimentation. Besides the BonFIRE and OpenLab testbeds, many other testbeds have been federated to form a large federation of more than 20 testbeds at the time writing.

Beyond the initial design deployed and validated within the BonFIRE project, the generic monitoring architecture design was introduced just prior to the start of the Fed4FIRE project [94]. Fed4FIRE has been taken in this thesis as a reference federation in terms of requirements gathering, implementation, deployment and validation of the final MAFIA architecture.

Analog to MAFIA, the Fed4FIRE federation architecture follows a heterogeneous federation approach, where the main actors in the federation interact via common APIs while keeping their monitoring and management tools.

The Fed4FIRE monitoring system fully follows the architectural design principles of MAFIA and adopts its services presented in Sec. 4.3.1 [143]. Concerning the implementation discussed in Sec. 6.2, only some of the main functional elements presented in Sec. 6.2.1 are implemented. Each testbed uses its own local monitoring tools and most of them follow the recommended tools that are commonly used across testbeds (e.g. Zabbix, Nagios, Collectd). OML is used as a common API. OML adapters are used to provide monitoring data following the OMSP format for some services that are implemented, as not all testbeds support all the monitoring services discussed in Sec. 4.3.1 at the time of writing. Concerning the semantic support, the use of the MOFI and OMN ontologies, and user-friendly data access for visualization, these are planned for the final cycle of the project, which has just begun.

Only some of the implemented services presented in Sec. 6.2.2 have been adopted in the Fed4FIRE monitoring system so far. A health and status monitoring service for FLS has been implemented, named facility monitoring in Fed4FIRE. High-level monitoring information about all federated Fed4FIRE testbeds is displayed through the Fed4FIRE FLS monitoring dashboard.[10] Some testbeds has implemented infrastructure resource monitoring service for federation services, such as SLA and reputation. Concerning the infrastructure resource monitoring service for users (Fed4FIRE experimenters), only the user-friendly approach has been adopted. It is implemented by around half of the testbeds to date. One of them, the FUSECO PG, adopted the implementation provided in Sec. 6.2.2.1 as is. Nevertheless, all testbeds will adopt these services and their implementations in the final cycle of the project.

[10] https://flsmonitor.fed4fire.eu

7.1.6. The FP7 ICT Infinity Project

One of the MAFIA monitoring services, namely the high-level infrastructure health and status monitoring service presented in Sec. 4.3.1.1, has been adopted within the INFINITY project. This service bas been integrated within the XiPi (`xipi.eu`) portal to provide information for various user communities about European FI testbeds besides diverse kinds of information of their capabilities, locations, etc. This service is only visible for XiPi's registered users, as per XiPi's sustainability strategy.

Figure 7.7 shows the implementation of the health monitoring service. The implementation comprises two central main components integrated with the XiPi platform: the monitoring module and the monitoring frontend (GUI) [170].

The implementation of the monitoring module follows the implementation presented in Sec. 6.2.2.2. Registered infrastructures push monitoring data as OML streams to this module, which offers an OMSP interface. The module stores the data with the corresponding infrastructures in own data storage. It gets a list of the already registered infrastructures via the REST interface provided by XiPi portal.

The monitoring frontend has been integrated with the environment used by the XiPi portal (Liferay). It is implemented as two integrated portlets and a Web service. It pulls the status information from a REST interface offered by the monitoring module to display results. More details are found in [170].

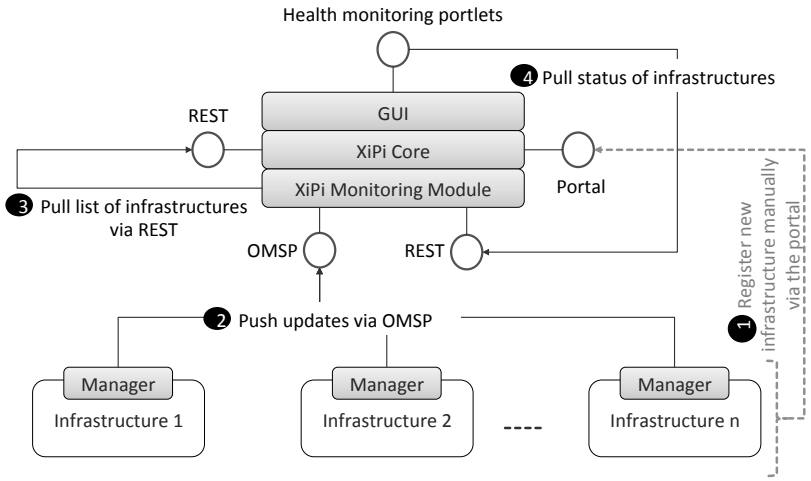


Fig. 7.7.: XiPi health monitoring service implementation [170]

7.1.7. Fraunhofer FUSECO Playground

FUSECO PG is built and operated by the Fraunhofer Institute FOKUS and TU Berlin, and is a standalone testbed that is also part of the Fed4FIRE federation. The reference implementation of MAFIA presented in Sec. 6.2 is used by FUSECO PG to provide the monitoring services supported by MAFIA for Fed4FIRE users.

7.2. Experimental Evaluation

This section discusses the validation of the usability and effectiveness of the designed solutions using the *controlled experiment* method according to Hevner et al. classification [160]. The aim of this evaluation is not to conduct a complex experiment but rather to demonstrate the usability and effectiveness of MAFIA through the deployment of a simple UCRE on a federation of testbeds. Such a UCRE is usually created by the user to conduct experiments to study something. Note that the term 'experiment' is used in this section in two different contexts; one refers to the demonstration of a controlled experiment to evaluate MAFIA, and the other refers to the experiment conducted by the user within the created UCRE.

Two testbeds were used for this evaluation. One is the Monitoring Demo Testbed located at the TU Berlin, and the other one is the FUSECO PG. Both testbeds have an OpenStack environment and use FITeagle as an infrastructure manager for managing (advertisement, provisioning and termination) their resources.

On each testbed, a VM was deployed, using jFed tool, which acts as an SFA client to the FITeagle's SFA API. Through jFed, sequential commands were sent to each of the two FITeagle instances to perform the allocation and provisioning the VMs individually. Upon receipt of these requests, FITeagle interacted with OpenStack to create a VM instance according to the requested image (see `sliver_type` in Listing 6.8). Both VM instances were created with an Ubuntu-64bit image, while each had different instance sizes, a tiny OpenStack flavor (512MB Random-Access Memory (RAM), 1 Virtual CPU (vCPU) and 5GB disk) at the Monitoring Demo Testbed and a medium OpenStack flavor (4GB RAM, 2 vCPU and an a 40GB Disk) at FUSECO PG. The latter was used as a server, while the the former acted as a client and downloaded some files from the server.

Both VMs along with their hosting PMs were monitored at testbed level. The Zabbix tool is used in both testbeds to monitor their infrastructures, each with its own deployment. In order to simplify the scenario, both testbeds uses Zabbix as a local monitoring tool, although different tools can be deployed.

As discussed in Sec. 6.2.2.1, FITeagle is used and extended in this work to implement part of the MAFIA functionalities (Figure 6.2). This version was running on the Monitoring Demo Testbed. After the provisioning of the VM, the monitoring module of FITeagle interacted with OpenStack to get the ID of the PM hosting the created VM. It then configured a Python-based OML wrapper to collect data from Zabbix about the VM and its hosting PM and export it as OML streams following RDF-based schemas. The wrapper performed this job in a regular basis, with an update rate of 10 seconds. An example of such a wrapper along with some of the used measurement metrics is shown in Listing C.1 in Appendix C. The same procedure was applied at the FUSECO PG, but different software was used to configure the wrapper.

Monitoring data received from both testbeds was collected in a collection resource created prior to the VMs. This resource was created as part of this experiment as an additional VM at the the Monitoring Demo Testbed. A special image was used to boot the VM instance that contains the semantic OML server and Jena Toolkit (Fuseki and TDB) as its backend. Its URI was given in the RSpec requests (see http://130.149.22.139:3030 in Listing 6.9) when creating the VMs. Related monitoring data was then pushed by the wrappers at the testbeds to the server.

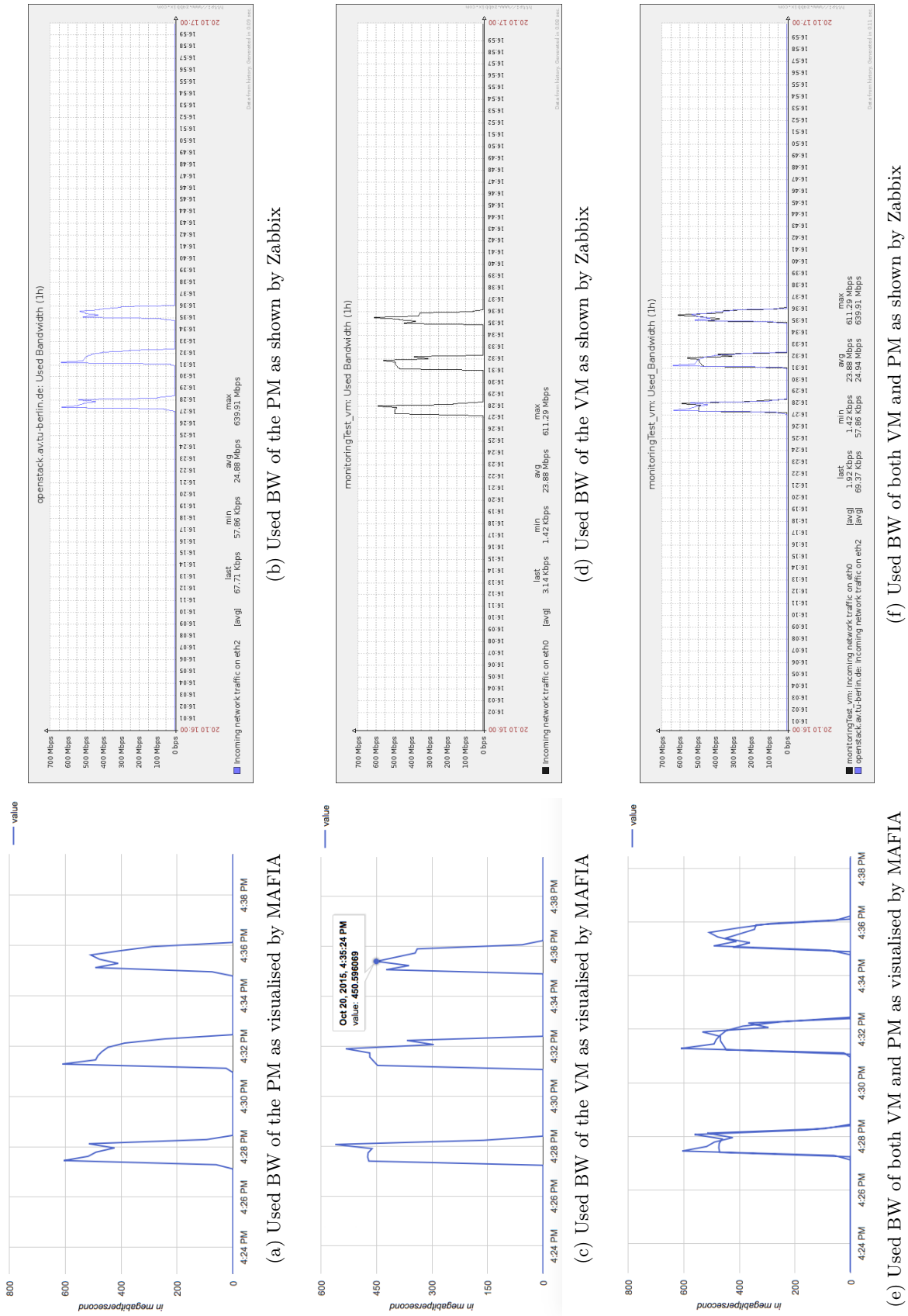
To demonstrate the effectiveness and reliability of MAFIA, the following activity was performed. The VM created to act as a client was used to download a file of 761 MB size from the VM deployed at FUSECO PG, which acts as a server. This file was downloaded 5 times consecutively and the same action was repeated three times with a four-minute rest in between. The bandwidth used by the client VM and its hosting PM are shown in Figure 7.8. The figure shows in the left side three line charts representing the bandwidth used by the PM (7.8a), VM (7.8c), and by both (7.8e) as shown by the visualization capability of MAFIA. The same information as originally shown through Zabbix are represented in 7.8b, 7.8d and 7.8f respectively. The figure shows that the behaviour of the VM is reflected by its hosting PM as expected.

However, the key message of this figure is to show that the monitoring data provided to the user through the visualization capability of MAFIA is the same as that originally shown through Zabbix at testbed level. This means that the user will get data in a common format after being converted at federation level, as if it were provided directly by the tool deployed at testbed level. The visualization capability of MAFIA can display the data in various chart types as responses to SPARQL queries sent over HTTP to the SPARQL endpoint of the Fuseki server. For instance, line charts are shown in the figure. Through these charts, more detail for a single measure, such as the date, human readable timestamp, and the exact measured value, can be displayed as shown in the subfigure 7.8c. Sample SPARQL queries are given in Appendix D.

Monitoring information about the UCRE and infrastructure resources was provided by the testbeds through the Python-based OML wrapper that was configured by the monitoring module of FITeagle. In order to demonstrate a user-related measurement, another OML wrapper was deployed on the client VM that measures packet delay (as shown in Figure 7.9) and loss on its link with the other VM that acted as a server. This wrapper used Ping to conduct measurements and exported data to the collection resource. It is written in C in order to demonstrate the flexibility of MAFIA in terms of smooth integration and multi-language support.

Monitoring data was semantically stored in graphs linked to their associated resources and testbeds, which are in turn linked to other information. Figure 7.10 shows an example of an RDF graph represented through LodLive[11], a tool used to browse RDF resources from any triple store using Linked Data standards (RDF,

[11] http://en.lodlive.it



(a) Used BW of the PM as visualised by MAFIA

(b) Used BW of the PM as shown by Zabbix

(c) Used BW of the VM as visualised by MAFIA

(d) Used BW of the VM as shown by Zabbix

(e) Used BW of both VM and PM as visualised by MAFIA

(f) Used BW of both VM and PM as shown by Zabbix

Fig. 7.8.: Used bandwidth of a VM and its hosting PM during performing a controlled experiment to evaluate the effectiveness and reliability of MAFIA

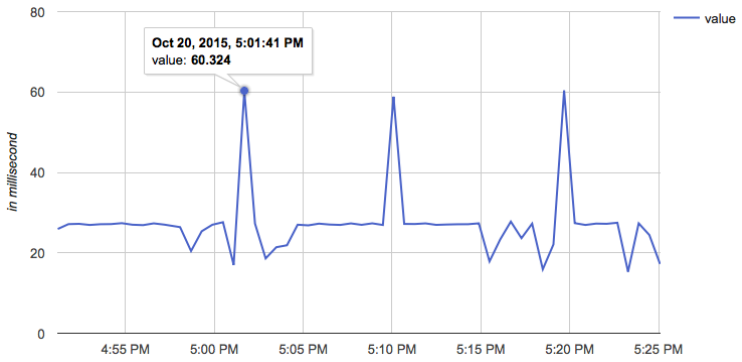


Fig. 7.9.: Packet delay between two VMs visualized by MAFIA

SPARQL). This simplified graph represents linked information around the client VM deployed in the controlled experiment. The information displayed in the list to the right of Figure 7.10 represents a simple measurement of `used_bandwidth` on the VM, following the schema used by the Python wrapper and represented in Listing 6.6 and Listing 6.7. This simple measurement has a data value of `0.002190` with the unit `gigabps` and a timestamp `2015-10-20 23:22:34+02:00`. The graph is further described in Sec. 7.3.2.

7.3. Analytical Evaluation

This section presents analytical evaluations of the solutions developed within this thesis. As this thesis focuses mainly on the unification of monitoring interfaces to provide data in a common and meaningful manner, this section focuses on evaluating a certain number of directly related components. That means components used to describe, discover and provision resources that are implemented by other tools, such as Teagle, FITeagle or any other tool (see Sec. 6.2.1), are out of the scope of this evaluation. In the reference implementation, the OML framework, related components like resource adapters (wrappers), and MOFI models represent the main functional elements of MAFIA. These are used to harmonize monitoring data and deliver it in a common manner according to shared conceptualisation and definitions among the various consumers. Therefore, it's especially worth evaluating the implementation of these components.

Several evaluation methods are discussed, including quality and correctness, effectiveness, performance, and environmental impact analysis.

7.3.1. Quality and Correctness Evaluation

The quality and correctness of the tools and models used within MAFIA are briefly discussed.

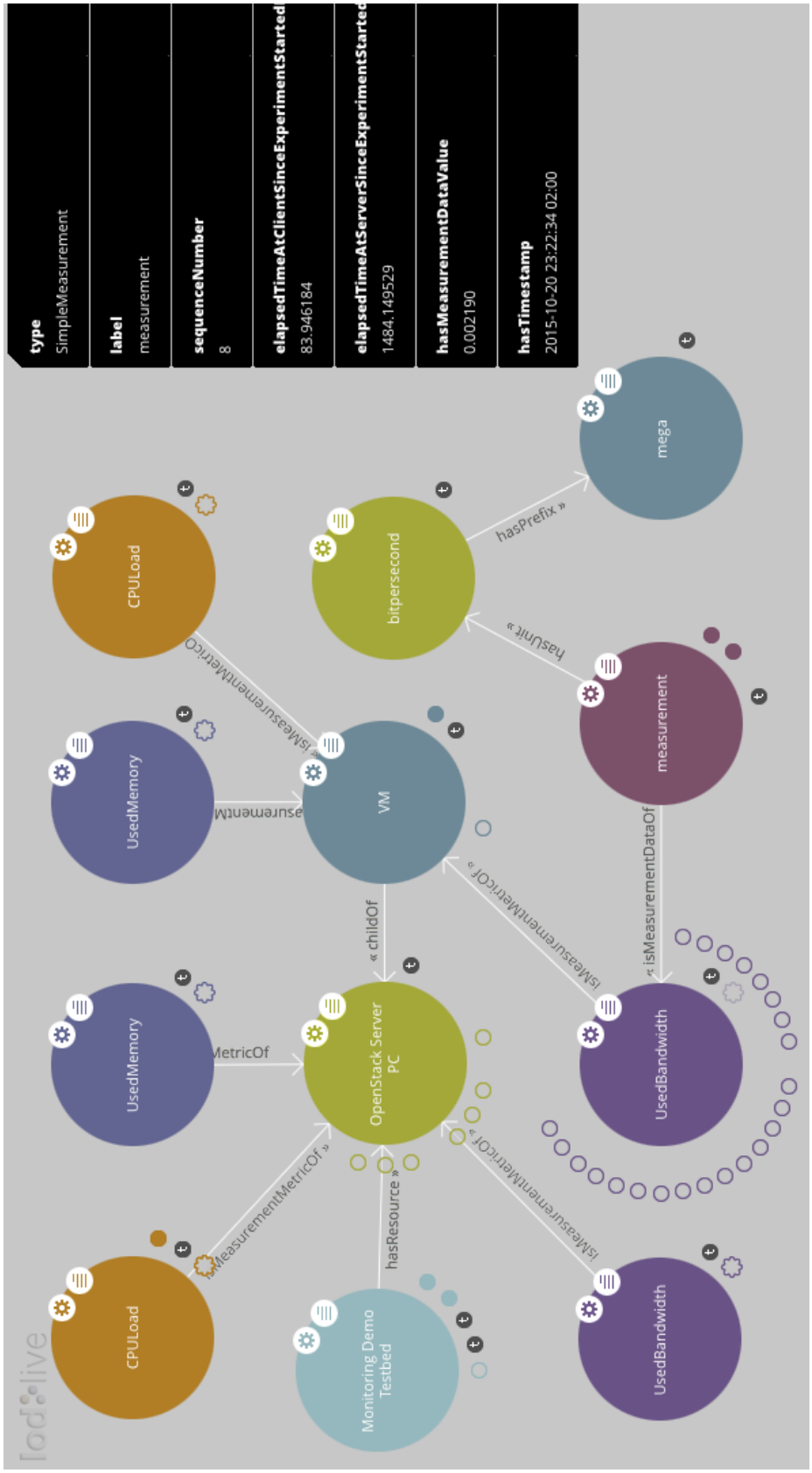


Fig. 7.10.: An RDF graph representation in Lodlive

As monitoring tools used for data acquisition are out of scope, the quality and correctness of the data produced by such tools are not discussed here. However, measured data and their original timestamps are made available for MAFIA users in the form originally provided by such tools. It is assumed these tools have a high level of accuracy and correctness.

As mentioned, the OML framework, MOFI and monitoring wrappers are the main components that are evaluated. OML per se takes care of minimizing side effects introduced by measurement, supports robust timestamping across measured devices, and is dynamically reconfigurable [123]. The semantic OML extension, even with the adoption of MOFI ontologies, is implemented in compliance with OMSP specifications and is still capable of collecting and processing monitoring data in both modes: semantic and classic data streaming. OML wrappers retrieve data from their sources and provided it as OML streams following OMSP specification and MOFI models. From this perspective, the correct use of OMSP and MOFI is essential.

MOFI ontologies are developed following best practice methods to ensure high quality and correctness. For this purpose, multiple considerations are taken into account (Sec. 5.2.1):

- The ontologies are developed following a standard methodology, which takes into consideration taxonomic precision, reusability and concept comparisons.
- MOFI ontologies are built using Protégé[12], which is a widely used ontology editor and knowledge-base framework. Furthermore, because of its expressive power, OWL 2 Web Ontology Language[13] (OWL2) is used for encoding the ontologies. For data serialization, RDF/XML and Turtle are used.
- Prior to the development of MOFI, relevant and existing ontologies were investigated and carefully evaluated in order to avoid redesign of the same subject matter and to reuse as much as possible from existing ontologies. Reuse is actually one of the main advantages of ontologies. Accordingly, significant concepts, relations and vocabularies were reused by MOFI from existing ontologies.
- During the development of MOFI, any change or update was automatically validated using the Protégé reasoner and Apache Jena Eyeball[14] inspectors, after having been manually validated via the OntOlogy Pitfall Scanner[15] (OOPS). Some validation steps can also be done during the use of the ontologies. For instance, while creating user schemas, the extended OML Scaffold program is used to validate their correctness and, in case of a typo for instance, a warning is declared and the most probable correct term is suggested as an alternative.

[12] http://protege.stanford.edu
[13] http://www.w3.org/TR/owl-overview/
[14] https://jena.apache.org/documentation/tools/eyeball-getting-started.html
[15] http://www.oeg-upm.net/oops

7.3.2. Effectiveness Evaluation

The use of a common API accompanied by well-defined information and data models overcome the heterogeneity of the interfaces, data models (e.g. tree or list based) and data formats (e.g. XML, JSON, CSV and DSV) of the various tools used in federated infrastructures. It is otherwise painful for users or applications to understand or process data collected from various sources. They would need to translate as many formats as are available in addition to parsing various data structures.

For unification of monitoring interfaces, the OMSP protocol is used in the prototype implementation of MAFIA in order to allow the transportation and collection of data in unified manner as lists of DSV. OMSP also allows arbitrary data schemas to be defined on-demand. In addition, MOFI ontologies are used in order to enable the creation of understandable, flexible data schemas without the need for them to be predefined or unique across domains. A short explanation of the effectiveness of the use of OMSP with MOFI ontologies is given as follows.

XML-based solutions are currently used to overcome heterogeneity issues in the domains addressed in this thesis. As discussed in [159], although XML schemas are extensible, their tree-based data structures do not allow explicit semantics. Furthermore, there are no rules for restricting which extension is permitted and in which part of the tree. As a result, syntax checking to parse the information cannot be easily applied. This is due to the fact that, every time a new extension is added, the checking process is left to the code parser, which has to be modified to cope with the new extensions. Such solutions will not scale in large and dynamically changing environments. In contrast, parsing (syntax checking) RDF triples is much easier. Further modifications are not required when additional information is included, as these are simply further RDF triples that can be handled as normal.

Furthermore, XML supports syntactic interoperability, but it supports neither semantics nor reasoning over the information. In contrast, semantic annotation and representation of data as RDF triples facilitates semantic interoperability. RDF identifies concepts, also called resources, with specific and unique URIs. This method allows all functional elements involved in a solution that uses semantic data representation to share the same knowledge. Via linked data capability, more information can be linked to form a graph of a large number of facts and relations as long as they are related to a given URI. This facilitates information discovery and mapping.

For example, SLA management is one of the federation services that consumes monitoring data provided by MAFIA. If all resources are identified by unique URIs, then all required information (e.g. resource ID, resource related monitoring information, and resource lifetime) is linked together as graphs, which can then be used for SLA validation, even if the information is stored in multiple locations. If the relations between UCRE resources (e.g. VMs), their related infrastructure resources (e.g. host PMs), measured metrics (e.g. CPU load), and SLA ID are all expressed via

RDF triples, the SLA management system can easily discover and find the relevant information required to validate SLAs.

Such capability significantly reduces time and complexity otherwise spent in the implementation without the semantics. This is reflected in practical experience within the Fed4FIRE project [159], as all involved parties had to agree on a specific SQL data structure (e.g. number of databases, tables, fields) for each service being handled. A further advantages of storing data triples with unique URIs is that it allows possible failures to be predicted vial liked data and as a result these failures can be handled proactively.

Figure 7.10 shows an example of an RDF graph representing information related to a user VM, which is an instance of *omn:VM* created as described in the controlled experiment discussed in Sec. 7.2. The infrastructure (`Monitoring Demo Testbed`) offers this resource, which is hosted on a PM, named `OpenStack Server`, which is in turn an instance of the class *omn:PC*. Further monitoring information about both the VM and its hosting PM are also provided, such as the `UsedMemory` and `CPULoad` as indicated in the graph. Each of the circles displayed in the graph shown in Figure 7.10 represents a resource. In this graph, all resources are represented via their labels, but are otherwise represented through their unique URIs. For example, the VM, PM and the Monitoring Demo Testbed infrastructure each have a unique URI. These are http://monitoring.service.tu-berlin.de/resource/Openstack-1/c277660a-f5c7-4c29-9fdb-590e6e57ecc2, http://monitoring.service.tu-berlin.de and http://openstack.av.tu-berlin.de, respectively, also represented in Listing 6.6.

By clicking on the letter *t* beside each circle representing a specific instance (RDF resource), a new circle is displayed indicating the type of the instance according to the OMN ontology. The small, not filled circles surrounding a large circle represent its related concepts. By clicking on any of these, related information is displayed. By clicking on the small circle with lines located at the top right of each large circle, the black list of information is displayed. The user can view all the information linked together in a large graph. For simplifying the figure, this graph illustrates only a few relations. An extended graph with more linked information related to this controlled experiment is shown in Figure D.2 in Appendix D.

In addition to the other benefits gained from using semantics, semantic reasoning allows logical information to be inferred from asserted triples. For example, from a given property, it's possible to automatically know the type of the resource this property is related to, according to the type of its domain or range.

As a result, the semantic-based approach presented here does not only allow the data to be represented in unified and meaningful manner, but also enables linking, matching, searching and reasoning of resource related information collected from different data sources.

7.3.3. Performance Evaluation

This section presents a number of experimentations that evaluate the performance of the implementation of MAFIA. This evaluation studies the performance of MAFIA with a focus on fundamental functional elements like OML wrappers and the OMN-based semantic OML. The performance evaluation of OML per se is out of the scope of this thesis. Nevertheless, it does not significantly ($\boldsymbol{p < 0.05}$) impact the instrumented services or applications in terms of performance, data accuracy and precision, as evaluated in [123].

In the performance evaluation discussed in this section, multiple runs of 10 experiments were conducted to investigate the performance of the reference implementation of semantic versus classic OML. The aim of these experiments is to investigate the overhead costs of the use of semantic technologies.

With the same configuration environments, two OML wrappers (non-semantic and semantic-aligned) were used to collect data from Zabbix and transmit it as OML streams to an OML server. This server supports both semantic and non-semantic data collection. In each experiment, 10,000 measurement values of particular metrics and their associated resources were transmitted as OML streams from the wrapper to the server. These 10,000 transmissions were pushed on a regular basis with an update rate of 10 seconds. In five experiments, the data was transmitted by the non-semantic OML wrapper as classic OML streams handled by the server following the classic procedure and stored in a PostgreSQL backend. In the other five experiments, the data was sent by the semantic-aligned OML wrapper following RDF schemas, was handled semantically at the server, and was finally sent to Jena Fuseki server to be stored in Jena TDB. In both cases, each of the five experiments had a different number of metrics being measured, transmitted and stored, namely 1, 5, 10, 15 and 20. An example of the semantic-aligned OML wrapper with only three of these metrics is shown in Listing C.1 in Appendix C. After transmitting OML header information including the schemas (see Sec. 6.2.1.2), data was then sent as streams, where each metric and its related information was sent through one OML stream.

In each experiment, was the time spent at the following four stages investigated as performance indicators:

- $\boldsymbol{T_{injection}}$: time spent for data injection at the client,
- $\boldsymbol{T_{processing}}$: time spent processing data at the server,
- $\boldsymbol{T_{insertion}}$: time spent inserting the data in the database, and
- $\boldsymbol{T_{delay}}$: time between data injection and data processing, which includes transmission delay between the client and server and the queuing delay before all OML streams being processed, i.e. after the last stream of each push is ready to be processed.

It is worth mentioning at this stage that the data are sent along with their original timestamp. However, this evaluation was conducted to examine only the delay and transfer cost caused by the use of the semantics.

Table 7.1 shows the evaluation results of the 10 experiments (five representing the classic use of OML and the other five semantic-based). To obtain more accurate results, each experiment was conducted in several runs (at least three) with the same setup and configuration, with 10,000 streams transmitted each time. The values in Table 7.1 represent the average times (in milliseconds) spent to process one stream (including information about different numbers of metrics) in several stages in classic and semantic OML. Each time represents the average of 30,000 measured values, where values of three runs per experiment are considered.

Tab. 7.1.: Performance evaluation for semantic and classic use of MAFIA's common API (OML/OMSP) – Part 1

Number of Metrics	**OML Type**	$\boldsymbol{T_{injection}}$ (ms)	$\boldsymbol{T_{processing}}$ (ms)	$\boldsymbol{T_{insertion}}$ (ms)	$\boldsymbol{T_{delay}}$ (ms)
1	Classic	0.162	0.272	0.386	0.216
	Semantic	0.188	0.316	19.998	0.266
5	Classic	0.307	0.500	1.564	2.256
	Semantic	0.428	0.863	100.682	105.416
10	Classic	1.125	0.878	2.957	4.218
	Semantic	1.394	1.743	209.192	216.398
15	Classic	1.649	1.333	4.584	6.517
	Semantic	2.373	2.844	306.111	320.003
20	Classic	4.965	1.810	6.722	9.461
	Semantic	5.667	3.459	427.640	446.418

Table 7.1 shows that there is no big difference in $\boldsymbol{T_{injection}}$ and $\boldsymbol{T_{processing}}$ in both cases (classic and semantic handling), with slightly more time in the semantic case. This is because around 25 triples were processed to represent information about one metric, which is represented in the classic case through 9 key/value pairs.

However, the time spent for inserting ($\boldsymbol{T_{insertion}}$) the RDF triples into Jena Fuseki with a Jena TDB backend takes significantly longer compared to that for storing traditional OML data into PostgreSQL. This was expected, as the cost per unit information inserted in Jena TDB is much higher than relational database [171]. Furthermore, each stream was represented through 25 triples that were, with all their associated ontology prefixes, first processed at the Fuseki server and then stored in the TDB database.

Concerning the T_{delay}, including transmission and queueing delays, different results are obtained. For transmitting only one measurement metric in each stream, the delay is similar in both cases, with `0.216 ms` in classic OML and `0.266 ms` in semantic OML. This is because there was no queueing for OML streams, as only one stream was transmitted in each push. The deviation in the T_{delay} between the classic and semantic cases begins from 5 metrics upwards. This was induced by the queuing delays at server side, as streams were processed sequentially. Longer delays occurred in the semantic case, as the data processing and injection took longer. Although the T_{delay} values in Table 7.1 from 5 metrics upwards represent only transmission and queueing delays, they are the highest values. This is because each stream was delayed in a queue until all previous streams in the respective push were processed and their data inserted into the database. An example of how the queueing delay increased the T_{delay} in the case of 5 metrics is shown in Figure D.1 in Appendix D. The T_{delay} in the case of 5 metrics implicitly includes the time consumed for processing and inserting of 4 metrics (*N-1*, where *N* represents the number of metrics/streams). T_{delay} can thus be calculated as follows:

$$T_{delay} = transmission\_delay + \sum_{n=1}^{N-1} (T_{processing}(n) + T_{insertion}(n))$$

Similarly in the case of 10, 15 and 20 metrics, T_{delay} implicitly includes the time consumed for processing and inserting of 9, 14 and 19 metrics/streams respectively. Such delays are not significantly high in the case of classic versus semantic use of OML, as the time consumed for processing and inserting streams into PostgreSQL is low.

Although there are significant delays caused by data insertion into Jena RDF store, these are still within an acceptable range. This is because most, if not all, of the monitoring solutions provide data regularly with a regular update rate of 10 to 60 seconds. Thus, less than half a second to export a batch of data streams comprising more than 500 RDF triples, representing 20 different measurement metrics is still acceptable.

Nevertheless, the use of a faster RDF triple store would reduce the insertion time dramatically. According to [172], Virtuoso[16] is eight times faster than Jena Fuseki.

To give more information on the distribution of the obtained data, in addition to the mean values given in Table 7.1, Figure 7.11 shows the so-called five statistical numbers: minimum, first quartile, median, third quartile and maximum. These are represented in each individual subfigure in Figure 7.11, where each varies according to time ($T_{injection}$, $T_{insertion}$, $T_{processing}$ and T_{delay}) and the number of measurement metrics (1, 5, 10, 15, and 20). Note that the times in Figure 7.11 are represented along the vertical axis in microseconds on a logarithmic scale (base 10).

[16] `http://virtuoso.openlinksw.com`

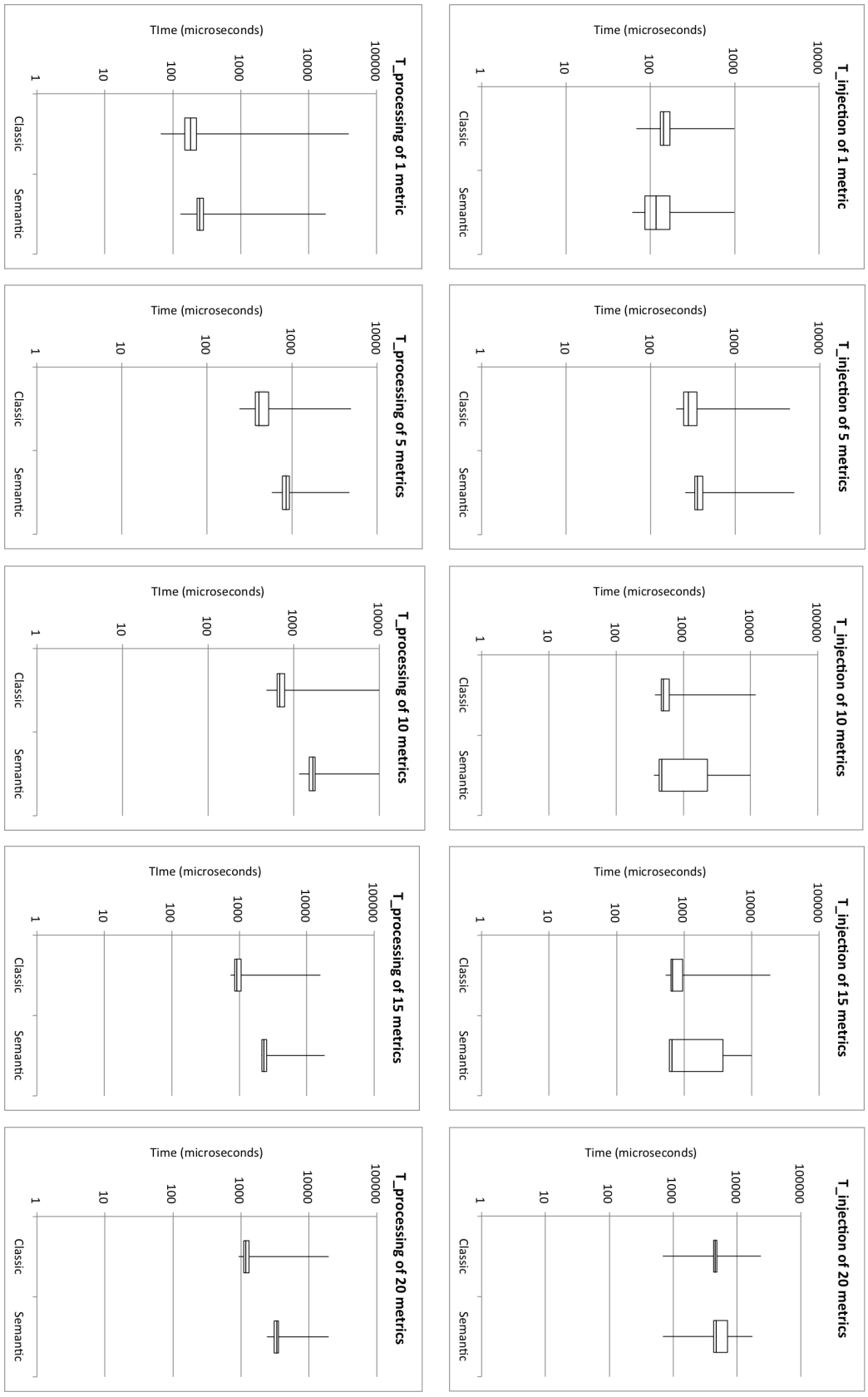


Figure 7.11, continued on next page ...

... continued from previous page

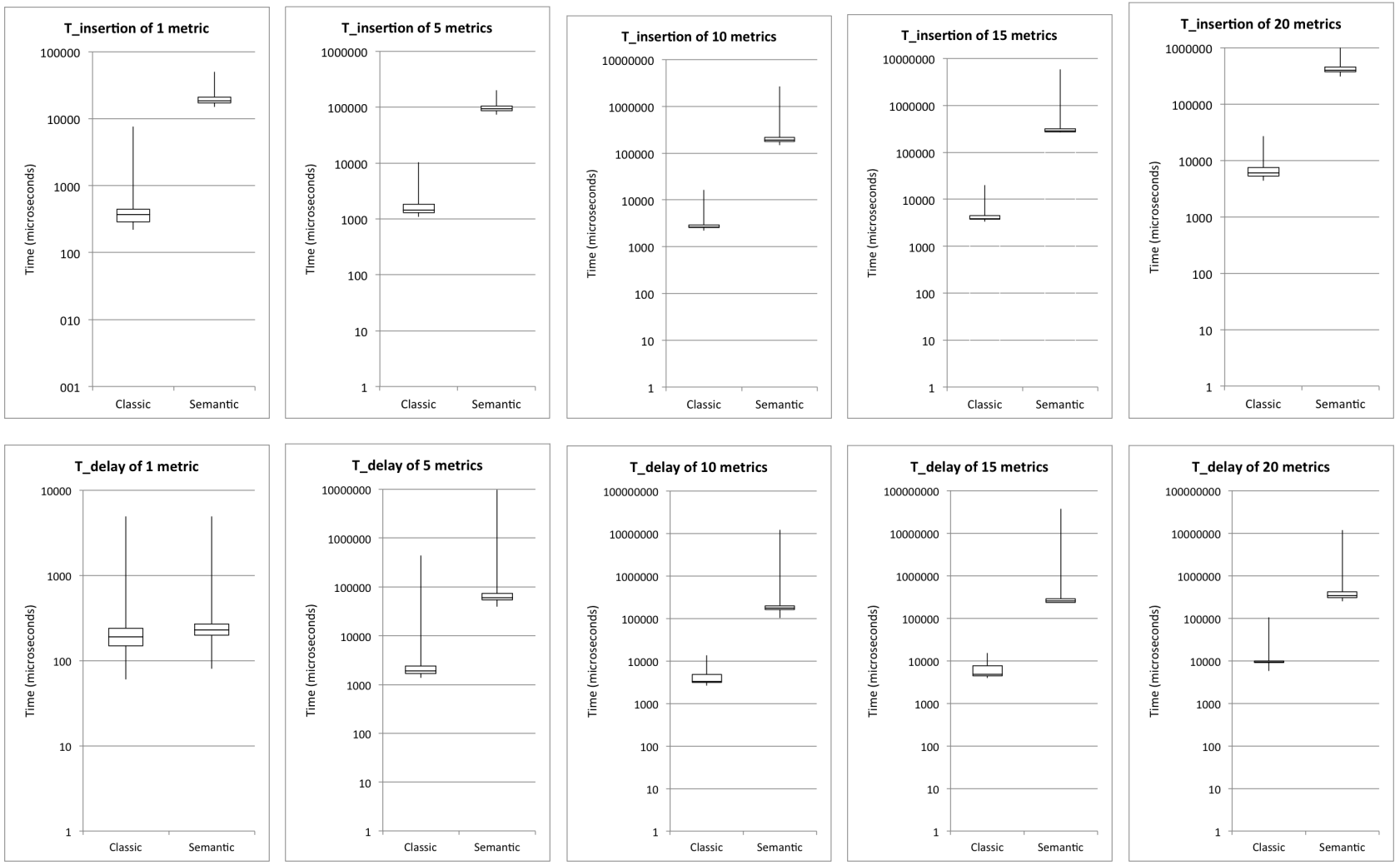


Fig. 7.11.: Performance evaluation results (statistical information (min, quartile 1, median, quartile 3, max) relates to the mean values shown in Table 7.1; times along the vertical axis are in base-10 logarithmic scale)

Further experiments were conducted to evaluate performance in the case of multiple clients (OML wrappers) simultaneously connected with the server and reporting data. In this evaluation study, five parallel OML wrappers running on different machines were connected with the server through different TCP sockets. Each wrapper reported data on about 10 measurement metrics with a regular update rate of 10 seconds. That means the server processes 50 metrics in each push. 1,000 pushes were performed by each wrapper to export data about the respective 10 metrics. This experiment scenario was executed twice: once in the case of classic use of OML and once in the semantic-based case. Furthermore, both experiment types were run multiple times. In this experimentation, only the performance at the server side was investigated. Therefore, only the time spent for data processing and insertion were considered. The average values of the time spent to classically or semantically process ($\boldsymbol{T_{processing}}$) 50 metrics/streams and insert the data into the respective database ($\boldsymbol{T_{insertion}}$) are shown in Table 7.2. The results show an expected distribution comparable to those shown in Table 7.1. Similar to the previous experimentation, Figure 7.12 shows statistical information.

Tab. 7.2.: Performance evaluation for semantic and classic use of MAFIA's common API (OML/OMSP) – Part 2

OML Type	$\boldsymbol{T_{processing}}$ (ms)	$\boldsymbol{T_{insertion}}$ (ms)
Classic	6.811	13.721
Semantic	9.402	928.440

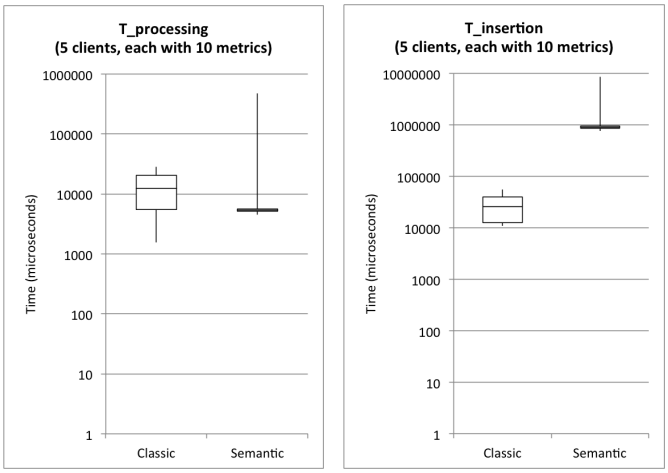


Fig. 7.12.: Performance evaluation results (statistical information (min, quartile 1, median, quartile 3, max) relates to the mean values shown in Table 7.2; times along the vertical axis are in base-10 logarithmic scale)

7.3.4. Impact Evaluation

The impact and overhead of the used monitoring tools should be kept to a minimum. In particular, when they are deployed on the same machines where other applications or services are running.

This evaluation investigates the impact of fundamental components of MAFIA in terms of resource consumption, such as CPU and bandwidth.

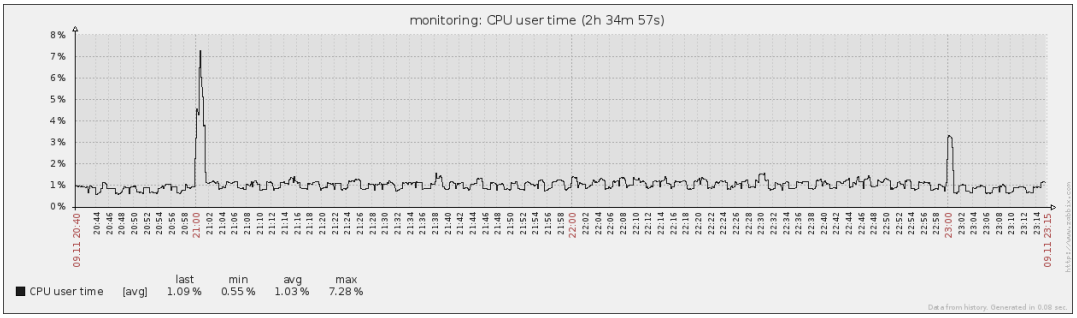
In this evaluation, two VMs were created in the Monitoring Demo Testbed. An infrastructure resource monitoring service was requested during setup. A Python-based OML wrapper was used to provide information about 10 measurement metrics per resource, collected from Zabbix. The wrapper fetched data from Zabbix and pushed it as OML streams, following RDF-based schemas to the semantic OML server. To investigate the impact of MAFIA in this case, only the Python wrapper and FITeagle on a Docker container were running on a machine. These then interact with other components, such as OpenStack, Zabbix and an OML server running on other machines.

Figure 7.13 shows the results of an experiment lasting 2 hours. The experiment started at 9:00 p.m. and stopped at 11:00 p.m.

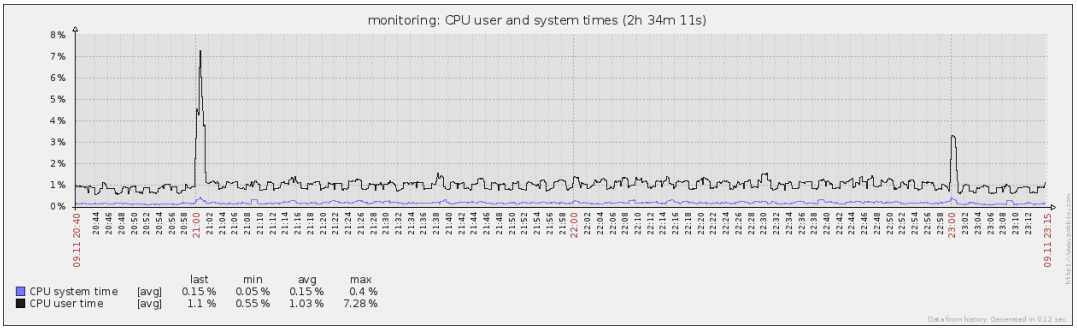
Graph 7.13a shows the CPU user time, represented in graph 7.13b in a black line, compared to the CPU system time represented in blue. Graph 7.13c shows further information about CPU utilization, where CPU use is almost negligible when compared to the CPU in an idle state (around 99% represented in red). The first jump in the first three subgraphs represent the execution of four successive SFA AM API calls sent by jFed and processed by FITeagle: *getCredential*, *ListResources*, *Allocate* and *Provision* for two VMs. The last jump represents the execution of the *Delete* call to remove both resources. However, it is clearly shown in the time period between the first jump and the last jump (i.e. from 9:00 p.m. to 11:00 p.m.), that the CPU use is very small at around 1%. Most of this usage (around 0.8%) is induced by other user processes (e.g. the Docker container), as shown in the first three subgraphs before the start time (9:00 p.m.) and after the stop time (11:00 p.m.) of the experiment.

Graph 7.13d in Figure 7.13 shows bandwidth utilization in Kbps. Black line represents the bandwidth used for incoming traffic (data retrieved from Zabbix), while blue line represents bandwidth used for outgoing traffic (data exportation to the semantic OML collection resource). The bandwidth used for incoming traffic is not an issue, if the wrapper and Zabbix run on the same machine, or enough bandwidth capacity in the local network is available.

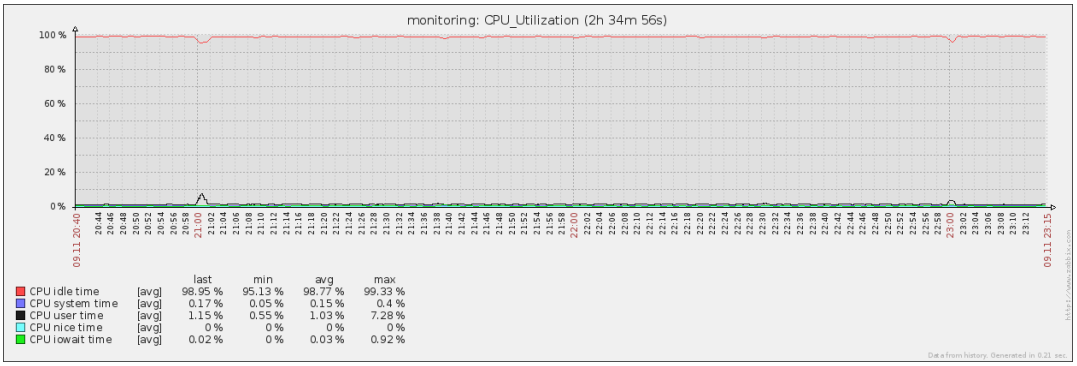
To be concluded that the impact of MAFIA components in particular the wrappers is almost negligible.



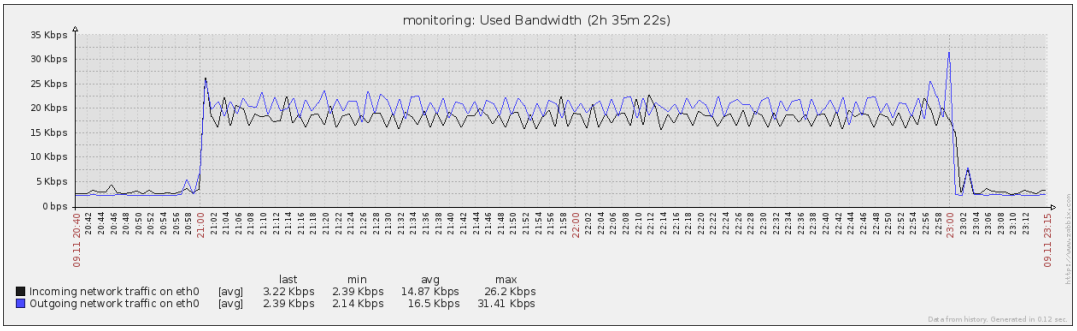
(a) CPU user



(b) CPU user and system times



(c) CPU utilization (idle, system, user, nice, and iowait times)



(d) Bandwidth utilization (incoming and outgoing traffic)

Fig. 7.13.: Impact of MAFIA wrapper in term of CPU usage and bandwidth during 12 hours of pushing monitoring data for two VMs

7.4. Requirements Validation

Chapter 3 presented and analyzed a wide range of requirements collected from different sources. This section analyzes the identified requirements against the specified and developed methods and components delivered by this thesis, in particular MAFIA and MOFI. Thus, it validates the fulfilment of the addressed requirements and presents the added values gained from the methods used.

Table 7.3 shows how the requirements identified in this thesis are fulfilled by MAFIA and its associated methods and models. Some requirements are either partially (e.g. standardisation and openness, and security) or indirectly (e.g. time sensitivity and accuracy) supported. Scalability is one of the requirements that is not addressed in this thesis. However, MAFIA is designed in a flexible way to scale so that scaling should be possible.

Tab. 7.3.: Identified requirements and methods specified and developed by this thesis

Requirement	**MAFIA Support**
Req-1 Cross-layer monitoring	Several monitoring services from low-level resources up to applications are supported.
Req-2 Cross-domain monitoring	The main focus of MAFIA is to operate cross-domain.
Req-3 On-demand	MAFIA services are offered on an on-demand basis.
Req-4 Autonomic	MAFIA is designed to support on-the-fly reconfiguration to adapt to the dynamic changes of the measured entities.
Req-5 Comprehensiveness	Any monitoring or measurement tool can be integrated. Thus, any entity that can be measured is integratable.
Req-6 Extensibility	MAFIA allows to add new measurement metrics at any level in the federation or even integrate additional monitoring tools.
Req-7 Scalability	Scalability has not been directly addressed.
Req-8 Standard and common data representation	This requirement is supported through standardized and common ontology-based information model (see MOFI models in Chapter 5).
Req-9 Programming interfaces	Data is provided via a common API (OML) and can be accessed through a standard SPARQL Query API.
Req-10 Standardization and openness	Partially supported. Data is provided via OML, which is open but not standardized, and can be accessed via standardized SPARQL Query API.
Req-11 Interoperability and compatibility	Supported by a common data collection and transportation protocol (OMSP) and a common information model (MOFI).
Req-12 Continuous monitoring	It depends on the implemented service; e.g. UCRE monitoring is provided for the interested consumers (the user or any federation service) continuously over the course of the UCRE's lifetime.

Table continued on next page ...

... Table 7.3 continued from previous page

Req-13 SLA monitoring	MAFIA provides capabilities to support SLAs across domains. It further delivers specification on how to provide monitoring data, but the type of measurement metrics are out of scope as they are application and domain specific.
Req-14 Time sensitivity	Not directly supported as it is the job of the tools used to perform measurements.
Req-15 Accuracy	Not directly supported as it is the job of the tools used to perform measurements.
Req-16 Reliability	Supported in that it provides all requested services and avoids single points of failure that may occur, as the architecture is designed to work in a distributed manner.
Req-17 Archivability	Data can be archived after the service or experiment lifetime.
Req-18 Security	Partially supported. Only authorized users are allowed to access or deal with the data.
Req-19 Minimal impact	Partially supported. MAFIA allows any tool to be used to produce data, and the impact of tools varies from one to the other. OML, as the main functional element in the implementation, has been designed and implemented to avoid additional overhead [123].
Req-20 Tool-agnostic	MAFIA is designed to be tool-agnostic, so that any monitoring tool can be used.
Req-21 Reusability	Adapters written for integrating tools can be reused by any other infrastructure that uses the same tool with limited reconfiguration efforts.
Req-22 Federation aware	MAFIA is designed to operate on a federation level and, thus, can interoperate with many other systems.
Req-23 Real-time and historical data	Both are supported.
Req-24 Data delivery	Infrastructures provide data to the user collection resource in a push manner, data is then accessed by the user in a pull manner.
Req-25 Ease of use	Once requested, users monitoring data is exported from multiple infrastructures to their own collection resources in a common way.
Req-26 Usefulness	Users can deploy different setups and get related monitoring data automatically if requested.
Req-27 Customizability	Users can define their own measurement metrics.
Req-28 User-friendliness	Partially supported, as it was not at the focus of MAFIA.
Req-29 Data storage flexibility	Users can create their own data collectors as VMs with flexible storage size. Infrastructures send user data to their collector or any collection endpoint identified by the user as long as it understands the protocol used (OMSP in the reference implementation of MAFIA).
Req-30 Data availability	Data is available for users during and after the lifetime of their created UCREs.
Req-31 Group support	Indirectly supported through granting privileges to the database to other users by the main user who owns the data collection resource.

7.5. Comparison with other Solutions

This section compares the achievements of this thesis with the other relevant state-of-the-art solutions introduced in Chapter 2. The most relevant solutions as summarized in Table 7.4 are considered in this comparison according to various criteria. The criteria are based on the requirements driving this thesis (discussed in Chapter 3) and are briefly described along with some remarks.

- **Cross-Layer Monitoring**: This type of monitoring is defined in Sec. 2.5.1.2. Taken as a criterion, it indicates whether the solution is capable of monitoring multiple layers, or only focuses on a particular layer.
- **Cross-Domain Monitoring**: This type of monitoring is defined in Sec. 2.5.1.3. Taken as a criterion, it indicates whether the solution can operate across domains. Usually, monitoring tools target homogeneous, single domains but not federated ones. Some tools supports some federation monitoring aspects, but these are very limited. For example, PerfSONAR measures network performance between domains (this thesis refers to such tools as inter-domain monitoring).
- **Federation Aware**: This criterion refers to whether the solution architecture is integratable with the target federation architecture and can interwork with the rest of the components of the architecture.
- **Data Representation**: This criterion refers to the methods, data formats and data structures used to represent monitoring data.
- **Information Model**: This criterion indicates whether the solution follows a specific information model to represent the data. This is the case for some solutions to avoid any possible confusion if information is represented differently. Such solutions are domain or implementation specific, e.g. FIWARE monitoring.
- **Interoperability and Compatibility**: This criterion indicates the ability of the solution to easily interwork with many other tools or solutions in an interoperable and compatible manner.
- **Tool Dependency**: This criterion indicates whether the solution depends on particular tools to support monitoring services, or is flexible to accommodate new tools (e.g. those in place in newly joined infrastructures) with minimum adaptation and as few changes as possible.
- **Extensibility**: This criterion relates to the possibility of extending the solution to include additional monitoring services or even monitor new areas or domains.
- **Customizability**: This criterion signals whether it's possible for users to define their own, customized measurement metrics.

Tab. 7.4.: Comparison of different monitoring solutions

Solution	**Cross-Layer Monitoring** (1)	**Cross-Domain Monitoring** (3)	**Federation Aware** (2)	**Data Representation** (1)	**Information Model** (3)
PerfSONAR	out of scope (-)	it focuses on inter-domain network performance monitoring (+)	supported, but limited to the network domain (+)	limited and specific to the tools used and their associated protocols (e.g. OWAMP) (+)	based on OGF specification and limited to network measurement metrics (+)
INSTOOLS	network measurement with limited OS measurements (+)	supported, but limited to particular domains (distributed servers and network performance measurements) (+)	supported but limited to the GENI federation (+)	SNMP and Secure Shell (SSH)/Secure Copy (SCP) (+)	SNMP's MIB (+)
FIWARE Monitoring	supported (++)	supported but limited to cloud infrastructures (+)	supported (++)	following the OMA NGSI specification (++)	limited to a list of datasets (-)
RESERVOIR Monitoring	supported (++)	supported but limited to cloud infrastructures and supports only three probes (CPU, memory and network usage) (+)	supported but limited to cloud domain (+)	there is no information on this (0)	own model that uses Distributed Hash Table (-)
TopHat	not supported, but still possible through the integration of suitable tools (+)	supported but limited topology monitoring and network measurements (+)	works on a federation but with no direct interoperability with other systems (e.g. control) (+)	XML-based or visualized through MySlice (+)	limited requests to query its Extensible Markup Language - Remote Procedure Call (XML-RPC) API (+)
MINER	out of scope (-)	out of scope (-)	out of scope (-)	XML documents (++)	based on the W3C XML schema (+)
Zabbix	supported (++)	possible for particular domains only (networks and distributed computer systems) (+)	out of scope (-)	JSON format (++)	own model (-)
Nagios	supported through third-party plugins (++)	same as Zabbix (+)	out of scope (-)	graph-based, or as JSON or XML through plugins (++)	Round Robin Database (RRD) files (-)
MAFIA	**supported** (++)	**supported** (++)	**one of its main architectural design principles** (++)	**through OMSP format along with an ontology-based data model** (++)	**MOFI ontologies** (++)

Table continued on next page ...

... *Table 7.4 continued from previous page*

Solution	**Interoperability and Compatibility** (2)	**Tool Dependency** (2)	**Extensibility** (1)	**Customizability** (1)	**Weighted Overall Score**
PerfSONAR	limited to the adopted standard tools (+)	depends on few tools (Ping, Iperf, Tracerout, etc.) (0)	yes, but only for new network metrics (+)	out of scope (-)	10
INSTOOLS	limited to the adopted standard tools (+)	depends on a set of tools (SNMP, tcpdump, NetFlow) (0)	possible, but requires extensions in through the whole software stack (+)	out of scope (-)	12
FIWARE Monitoring	limited to the used GEs (0)	tool-agnostic, currently based on Nagios and PerfSONAR (++)	supported (++)	supported through the used tools (+)	15
RESERVOIR Monitoring	limited to the adopted tools (0)	depends on a set of own probes (0)	out of scope (-)	out of scope (-)	2
TopHat	only supported between TopHat and the integrated tools (0)	it depends on the supported tools (through gateways, as discussed in Sec. 2.5.2.2), but can be extended (++)	extensible through gateways (++)	out of scope (-)	15
MINER	possible through proxies, to the tools supported, that provide data following XML schemas (+)	tool-agnostic but the available implementation covers a set of network measurement tools (++)	possible (+)	out of scope (-)	5
Zabbix	out of scope (-)	own implementation, but some standard tools (e.g. SNMP agents) can be integrated (0)	possible (+)	supported (+)	2
Nagios	out of scope (-)	like Zabbix (0)	possible (+)	possible through plugins (+)	2
MAFIA	**supported through a unified protocol (OMSP) and a common information model (MOFI)** (++)	**tool-agnostic** (++)	**supported through wrappers** (++)	**supported through wrappers and/or OMSP-aligned MPs** (++)	**32**

Criteria weights: (3) high importance, (2) medium importance, and (1) low importance — Rating: (++) accepted (2), (+) limited (1), (0) neutral (0), and (-) rejected (-1)

7.6. Summary

This chapter presented the evaluation methodologies that have been used to validate and investigate the performance of the solutions delivered within this thesis, in particular MAFIA and MOFI. According to the methodologies defined by Hevner et al. [160], three different evaluation methods have been undertaken.

The first is an observational evaluation method aimed at validating and tracking use, extensions and improvements of the delivered solutions within a set of practical, large-scale projects. Sec. 7.1 presented the main projects this thesis has contributed to and gained experience from. These projects have served to further enhance the delivered solutions that have been designed (Chapter 4, Chapter 5) and developed (Chapter 6) in an interactive, incremental manner.

The usability and reliability of MAFIA have been demonstrated through the experimental evaluation method discussed in Sec. 7.2. Besides demonstrating the usability and the cross-layer monitoring support of MAFIA, this evaluation has also shown that data provided to the user semantically, after being converted into the unified format at the federation level, is the same data as originally shown in the tool deployed at infrastructure level.

The third evaluation method is an analytical approach (Sec. 7.3) that investigated the quality, effectiveness and performance of the presented prototype implementation of MAFIA, as well as its environmental impact while running. In this evaluation, the effectiveness of the use of MAFIA was discussed, in particular in regard to the ontologies and their associated Semantic Web technologies. The added value gained includes linking information, conducting complex queries and reasoning over heterogeneous data, which allows information to be deduced without the need to extract it from different data structures. To check the overhead induced by the use of Semantic Web technologies within MAFIA, performance evaluation has been conducted. The results have shown that the time spent processing data at the server, in particular storing data in a triple store, takes longer compared with non-semantic data delivery, processing and storing in an SQL database. However, the induced delay is still within an acceptable range (in milliseconds), which does not have a noticeable effect. Finally, the evaluation study, that has been conducted to investigate the impact of MAFIA on its running environment, has shown that MAFIA has a very limited effect in terms of resource consumption (e.g. below 0.1% of CPU).

This chapter discussed the mapping of the identified requirements to the methods specified and developed within this thesis (Sec. 7.4) and presented a brief comparison between the solution delivered by this thesis and a set of the other most relevant solutions available (Sec. 7.5).

8 Conclusion

THE research of this thesis has focused on monitoring and measurement services across federated, heterogeneous ICT infrastructures. This is mainly due to the emerging needs and the lack of proper monitoring solutions that can operate across multiple administrative domains and deliver various and extensible sets of monitoring data in a common way within a heterogeneous federation, as discussed in Chapter 1.

The absence of a suitable existing solution along with the various and emerging requirements of multiple stakeholders from different sources (Chapter 3) have lead to the core contributions of this thesis. An extensive study of the state-of-the-art and research outcomes was conducted within this thesis (Chapter 2). This study served as a starting point for the main research results of this thesis, which are a monitoring architecture, an ontology-based information model and a prototype implementation, presented in Chapters 4, 5 and 6 respectively, and evaluated in Chapter 7. Together these provide a solution for the unification of monitoring interfaces of federated, heterogeneous ICT infrastructures, focusing on Cloud and FI experimentation areas, thus aggregating large, different and extensible sets of data from various sources.

This final chapter first presents a summary of the findings and the main contributions of this thesis. It then presents the dissemination activities, the impact of the achievements of this thesis, and its contributions to several research projects and standardization bodies.

8.1. Summary

The objective of this thesis was to design and develop a monitoring solution that operates across federated infrastructures and caters for collecting monitoring data provided by different monitoring interfaces, in a unified format following a common vocabulary. To this end, the following two main research questions were formulated as discussed in the introduction (Chapter 1):

1. *How to design an architecture that i) allows the integration and management of heterogeneous monitoring solutions distributed in a federated environment in order to provide a set of monitoring services in a common manner, and ii) is extensible to allow monitoring of other similar fields of application?*

2. *How to model heterogeneous monitoring and measurement related concepts and relationships allowing the target solution to provide the data in a common and meaningful way?*

Achieving the objectives of the thesis and answering these questions were at the heart of the research work performed during the course of this thesis. The main research results consist of three main contributions. These are briefly discussed and conclusions are drawn as follows:

1. The *monitoring architecture* for federated, heterogeneous infrastructures, called MAFIA, comprises a set of main components (Sec. 4.3.2): monitoring tools at infrastructure level, a common API, a set of adapters, data collection and storage, and data access and visualization. In answer to the first research question, MAFIA is designed in a generic way to support the main monitoring services required in a federation such as those addressed in this thesis. These include the federation of Cloud and FI testbed infrastructures, and the accommodation of any type of data collected from any data source as long as its design architectural principles (Sec. 4.2.1) are adhered to. In this thesis, seven different types of monitoring services are discussed (Sec. 4.3.1) and implemented (Sec. 6.2.2). Through an adaptation layer implemented by a set of adapters, a high level of abstraction of the diverse set of monitoring tools distributed in a federation enables the integration and management of data collected from various sources. MAFIA follows a federation model where each infrastructure keeps its own tools in place, but adapts to the common API specifications. Thus, MAFIA can be extended to monitor additional infrastructures joining the federation or even include new monitoring services through additional adapters. It is not necessary to write new adapters from scratch, but those already in use (examples are in Appendix C) can be re-used with minor modifications to fullfil their planned tasks in terms of providing the target data wrapped from the locally deployed tools.

2. The *ontology-based information model*, MOFI, describes related monitoring concepts and relations in federated, heterogeneous infrastructures at the conceptual and semantic levels. It comprises a vocabulary of terms and specifications of their meanings, and a set of definitions. It is designed to assist MAFIA in order to provide monitoring information in a common and meaningful way, where information originates from various sources with different ways of describing monitoring data. It has a hierarchical structure and consists of an upper ontology, which defines the fundamental concepts and relations, and subjacent ontologies, which reuse and specialize concepts and relations from the upper ontology (Sec. 5.2). Each subontology focuses on a particular aspect of the monitoring process, like measurement metrics, data, units, tools and further generic aspects. In answer to the second research question, this thesis has shown how these formal, canonical models allow mutual understanding of shared conceptualizations and definitions. This can overcome interoperability issues caused by the use of syntactic data models, like tree- or list-based data models with arbitrary schemas, which lack consistent, standardized vocabularies and hinder tool interoperability and data consolidation. Furthermore, ontologies allow information to be linked, heterogeneous data to be reasoned and queried over, and information to be deduced without the need to be extracted from different data structures.

3. The *prototype implementation* of the monitoring architecture (MAFIA) contains an implementation of the ontology-based information model (MOFI). This implementation covers the main components of MAFIA, namely the monitoring tools at infrastructure level (Sec. 6.2.1.1), a common API (Sec. 6.2.1.2), a set of adapters (Sec. 6.2.1.3), data collection and storage (Sec. 6.2.1.4), and data access and visualization (Sec. 6.2.1.5). The usability and effectiveness of the solutions delivered by this thesis have been demonstrated in Sec. 7.2 and Sec. 7.3.2. It can therefore be said that MAFIA is flexible enough to integrate and manage heterogeneous monitoring solutions distributed in a federated environment. This serves to provide various sets of data in a common and meaningful manner. MAFIA is furthermore extensible through adapters to allow monitoring of other similar fields of application. The analytical evaluation results have shown that the use of semantic technologies induces some delays, in particular during insertion of data into triple stores (Sec. 7.3.3). However, it was suggested in this thesis to use other faster triple store engines rather than the one used in the implementation. Nevertheless, it was observed that the semantic-based approach taken is beneficial for collecting and representing monitoring data in a flexible, unified and meaningful manner. The semantic-based approach also enables linking, matching, searching and reasoning of resource related

information collected from different data sources in the application field of FI experimentation across federated testbeds, which was not otherwise possible.

The design and implementation of the solutions delivered by this thesis have been undertaken in an iterative, incremental manner. They have been extended and improved after each iteration according to lessons learned from practical deployments. The solutions have been validated in large-scale environments, notably BonFIRE (a federation of seven cloud infrastructures) and Fed4FIRE (a federation of over 20 heterogeneous ICT infrastructures).

8.2. Dissemination and Impact

This section describes the dissemination activities that have been conducted during the course of this thesis, along with its academic and industrial impact. The academic impact is evaluated by the publication of the research results at the international highly ranked Association for Computing Machinery (ACM), IEEE and Informationstechnische Gesellschaft im VDE (ITG) conferences and workshops. The academic impact is further demonstrated by contribution to the teaching profile of the Chair of NGN (called "Architekturen der Vermittlungsknoten (AV)"[1] in German) at the Technische Universität Berlin. An indication of the industrial impact is provided through the use of the thesis artifacts in several industry driven testbeds and research projects. This is facilitated through the close and strong cooperation between AV and the Next Generation Network Infrastructures (NGNI) Competence Centre[2] at FOKUS Fraunhofer Institute for applied science.

The dissemination and impact of the outcomes of the thesis research are briefly listed as follows:

Publications: The publication track record of the thesis author includes 24 scientific publications between 2011 and 2015. These are listed in Appendix A and consist of one journal article, one book chapter, 18 conference papers, and four workshop papers.

Contribution to the scientific community: The expertise of the thesis author in the research fields addressed within this thesis has enabled him to further contribute to the scientific community in various ways. This includes reviewing a set of scientific papers, and the organization and co-chairing of three workshops. The author was the Chair of the International Workshop on UsiNg and building ClOud Testbeds (UNICO)[3], which was held in conjunction with the 5th IEEE International Conference

[1] http://av.tu-berlin.de
[2] https://www.fokus.fraunhofer.de/ngni
[3] http://www.bonfire-project.eu/unicocloudcom

on Cloud Computing Technology and Services in Bristol, United Kingdom, on 2 to 5 December 2013. The author participated in the organization of all aspects of the workshop, from workshop proposal formulation, through organization of call distribution, submission, review and acceptation processes, up to hosting and chairing the event. He was one of the organization committee of the 4th International IEEE Workshop on Open NGN and IMS Testbeds (ONIT)[4], which was held in conjunction with the IEEE Global Communications Conference (GLOBECOM) in Anaheim, California, US, from 3 to 7 December 2012. The author was the organizing chair for the 1st International IEEE Workshop on Multimedia Communications (MMCOM) 2011[5], held in conjunction with the 54th Annual IEEE GLOBECOM in Houston, Texas, US, from 5 to 9 December 2011. Finally, the thesis author has been contributing to and administrating the Free Tools for Future Internet Research and Experimentation (Free T-Rex) Platform[6], which provides access and knowledge dissemination for scientists, testbed operators and tool developers for tools and testbeds that are widely used in the area of FI experimentation.

Teaching contribution: The work within this thesis has contributed to the teaching profile of the AV Chair through enriching three master's degree courses with novel, relevant topics, where students are enabled to stay up to date with the current state-of-the-art. The thesis author has been an assistant lecturer in the course " Future Internet Technologies (FIT)" and supervisor of the student seminar course "Hot Topics in NGN & FI" between 2011 and 2014. He was a supervisor of the student project course "NGN & FI Projects" in 2015. Two bachelor's theses have been supervised by the author.

Artifacts usage: As discussed in Sec. 7.1, different artifacts of this thesis have been adopted and deployed in multiple testbeds in several research projects within the context of the largest relevant initiatives in Europe, namely the FIRE projects (e.g. BonFIRE and Fed4FIRE) and the FI-PPP projects (e.g. INFINITY and FI-STAR). Besides the software components, the contribution also consists of leadership and influence on the design and specification of the respective architectures. Furthermore, based on lessons learned from the initial implementation and deployment of MAFIA, this thesis influenced and contributed to the design of the XIFI monitoring architecture. Specifically, it influenced XIFI monitoring architecture to be tool-agnostic and open to integrating any cloud infrastructure with tools in place that adapts a specific set of adaptation mechanisms to become part of the federation [149]. The artifacts delivered by this thesis have been integrated into the FUSECO PG, in particular

[4] http://www.onit-ws.org/2012
[5] http://www.fuseco-workshop.org/mmcom2011
[6] http://www.free-t-rex.net

in its OpenStack-based Cloud testbed and the Wireless Lab that is powered by the licensed OpenEPC toolkit. FUSECO PG and other testbeds offered by the aforementioned projects target not only research communities but also SMEs and innovative startups. This offers them the possibility of establishing early-trials of their products in real-world, controllable, and cost-effective environments with in-depth monitoring capabilities. Furthermore, some of the initial research outcomes of this thesis have been used in the BMBF G-Lab Deep[7] project [168], [169]. Finally, the Mobile Cloud Networking[8] [190] (MCN) MaaS architecture[173] is conceptually inspired by the initial monitoring architecture design delivered by this thesis.

Talks and demos: During the course of this thesis, the author has given various talks and demos at international events to create, disseminate and share knowledge. Ten presentations at IEEE conferences and workshops have been given between 2011 and 2015. Training sessions were given within the XIFI training series (physical meetings and Webinars). These were offered for various technology development communities interested in FIWARE technologies and offerings, e.g. the Webinar training entitled "FIWARE Lab Solution for Managing Resources & Services in a Cloud Federation"[9] was given by the thesis author for an audience of 72 participants. The author was invited to give talks within the context of GENI-FIRE international collaboration. In the GENI-FIRE Workshop 2015[10], the author gave two talks including a demo on the use of ontologies, in particular OMN, for supporting the whole experiment lifecycle in federated heterogeneous ICT testbeds, taking their use for monitoring services as a use case. Furthermore, a couple of demos have been given in multiple events, such as FIRE meetings and workshops, ITG events within the G-Lab initiative and FOKUS FUSECO Forums.[11]

Standardization: The ontology-based information model (i.e. MOFI), as one of the three main outcomes of this thesis, represents the monitoring part of the OMN standardization initiative that focuses on modeling federated ICT infrastructures. At the time of writing, the OMN monitoring ontology, consisting of the six monitoring ontologies described in Chapter 5, is the largest OMN ontology in terms of the number of concepts and relations described. To make this initiative more accessible to the international community, OMN was registered with a permanent identifier (https://w3id.org/omn) and namespace (http://prefix.cc/omn) and the W3C Federated Infrastructures Community Group[12] was established.

[7] http://www.g-lab-deep.de
[8] https://www.mobile-cloud-networking.eu
[9] https://www.youtube.com/watch?v=1TKU4KdDybQ&feature=youtu.be
[10] http://tinyurl.com/firegeni2015
[11] http://www.fuseco-forum.org
[12] https://www.w3.org/community/omn/

8.3. Outlook

This section outlines plans and possibilities to exploit the potential of the presented work, as well as extensions of the work driven by currently running and planed activities.

The outcomes of this research work can be applied in federated ICT environments. The exact motivations for increasing demand in and type of federations go beyond the scope of this work. However, demand for federations of heterogeneous infrastructures is steadily increasing. Academic users are looking for federated heterogeneous testbeds to study complex phenomena across domains in real-world environments. In the cloud industry, SMEs can federate their resources and services to complement each other and can then compete against leaders in the cloud business, such as Amazon Web Services, Google and Windows Azure.

Indeed, various federation architecture models have been defined in the literature as described in Sec. 2.4.1. One of the main lessons learned within this thesis is that a heterogeneous federation architecture model is the most convenient model for large-scale, dynamic and sustainable federations. This model allows infrastructures to keep using their management systems and uses a common APIs on top of these systems. This way enables infrastructures to easily and quickly join or leave the federation. Therefore, MAFIA has been designed following this model and a common API is used on top of monitoring systems, thus catering for unification of data collected from their heterogeneous interfaces. MAFIA can be used to deliver various sets of monitoring services in federated infrastructures. It can be used by a federator to monitor and control the entire federation in terms of health and performance, SLA management, advanced reservation, and so forth. It can also be used by infrastructure providers to provide partial monitoring information for their users on-demand about the infrastructure resource used . Such a service is recognised as providing significant added value for users; for example, in a cloud federation, service providers who deploy their services in the forms of IaaS or PaaS across domains can get monitoring information about the used resources and their environmental conditions.

To ensure the sustainability and reuse of MAFIA, several steps have been taken. MAFIA was designed after considering a wide range of requirements from various stakeholders (e.g. software developers, infrastructure providers and federators). It is modular, not domain or task specific and can be extended to cover new areas of application through additional wrappers. The development of such wrappers doesn't require much effort, as reference implementations are provided that can be modified and extended according to the target use. De-facto standard protocols (SFA and OML) are used in the delivered prototype implementation.

As stated before, the design of MAFIA has been done in iterations. Its initial design and implementation can still be used to monitor a homogenous cloud federation, as it assumes the same monitoring tools are used across the federated infrastructures.

In addition to MAFIA, a significant contribution of this thesis is the monitoring ontologies (MOFI) implemented within MAFIA. They have been developed in a generic way to be (extended if required and) implemented in any ICT environment, allowing data exchange across domains and tools in a common and meaningful manner. MOFI is extensible to allow more – even domain specific – concepts and relations to be modelled. For instance, according to interest raised by the cloud standardization bodies IEEE InterCloud and EGI Federated Cloud Task Force, MOFI models are planned to be extended to model missing concepts and relations to serve their need. Furthermore, the thesis author was invited to present the possibility of using the developed ontologies within the context of the European FP7 FIESTA[13] Project, which focuses on the IoT domain. It's feasible to reuse the relevant IoT concepts from existing ontologies (e.g. SSN) within MOFI and extend it to fulfill the need of IoT platforms.

The impact of MOFI at the time of writing is limited to the FI experimentation field, driven by strong communities like GENI and FIRE. It is also of interest in federated Clouds. However, MOFI is not limited to a particular project but aims to reach a broader range of communities. Therefore, several steps have been taken to promote MOFI to achieve the broadest possible dissemination. This includes the use of the Dublin Core (DC) and Vocabulary for Annotating Vocabulary Descriptions (VANN) to describe meta data about individual MOFI ontologies. MOFI ontologies were serialized in two formats RDF/XML and Turtle. The latter provides human and machine readable serializations. Furthermore, the MOFI ontology is part of OMN standardization activity as aforementioned.

Last but not least, although the comparison of MAFIA with other solutions discussed in Sec. 7.5 has shown its advanced support and added value, there are still some requirements that have been identified but not fully supported. These are indicated as "not directly supported" or "partially supported" in Table 7.3 and can be considered for future work. Furthermore, according to the performance evaluation results of the prototype implementation, the use of a faster RDF/SPARQL server/triple-store is recommended and is also considered to be a matter for future work.

[13] http://www.fiesta-iot.eu

Acronyms

ACM Association for Computing Machinery ... 172
AM Aggregate Manager ... 129
API Application Programming Interface ... 3
AV Architekturen der Vermittlungsknoten ... 172
BMBF German Bundesministerium für Bildung und Forschung ... 34
BonFIRE Building Service Testbeds on FIRE [17] ... 30
bps bit per second ... 107
BWCTL Bandwidth Test Controller[14] ... 49
CAMP Cloud Application Management for Platforms ... 27
CAPEX Capital Expenditure ... 20
CCRA Cloud Computing Reference Architecture ... 40
CDMI Cloud Data Management Interface [174] ... 27
CH Clearinghouse ... 129
CIMI Cloud Infrastructure Management Interface [175] ... 27
CoMo Continuous Monitoring [176] ... 48
CONFINE Community Networks Testbed for the Future Internet[15] ... 30
CPU Central Processing Unit ... 5
vCPU Virtual CPU ... 147
CREW Cognitive Radio Experimentation World[16] ... 30
CRUD Create-Read-Update-Delete ... 51
CSA Coordination and Support Action ... 30

[14] http://software.internet2.edu/bwctl/
[15] https://confine-project.eu
[16] http://www.crew-project.eu

CSV Comma Separated Values . . . 79

DC Dublin Core . . . 176

DFA DETER Federation Architecture [177] . . . 29

DIMES DIMES[17] [178] . . . 48

DMTF Distributed Management Task Force . . . 27

DSV Delimiter Separated Value . . . 120

EGI European Grid Infrastructure . . . 3

ESNet Energy Sciences Network[18] . . . 49

ETOMIC European Traffic Observatory Measurement Infrastructure [179], [180] 48

EXPERIMEDIA EXPERIMEDIA[19] . . . 30

Fed4FIRE Federation for FIRE[20] [11] . . . 30

FEDERICA Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures[21] . . . 30

FedSM Federated IT Service Management[22] [73] . . . 35

FI-PPP Future Internet Public Private Partnership[23] [181] . . . 2

FIND Future Internet Design[24] . . . 28

FIRE Office FIRE Office [182]

FIRE Future Internet Research and Experimentation[25] [65] . . . 2

FITeagle FITeagle[26] [183] . . . 42

FIWARE Lab FIWARE Lab[27] [19] . . . 33

FIWARE Future Internet Core Platform[28] [184] . . . 3

FI Future Internet . . . 2

FI-STAR Future Internet - Social and Technological Alignment for Healthcare[29] [185] . . . 33

[17] http://www.netdimes.org/new/
[18] https://www.es.net
[19] http://www.experimedia.eu
[20] http://www.fed4fire.eu
[21] http://www.fp7-federica.eu
[22] http://fedsm.eu
[23] http://fi-ppp.eu
[24] http://nets-find.net
[25] http://ict-fire.eu
[26] http://fiteagle.org
[27] http://lab.fi-ware.org
[28] http://fi-ware.org
[29] https://www.fi-star.eu/fi-star.html

FIT Future Internet Technologies ... 173

FLS First Level Support ... 11

FRCP Federated Resource Control Protocol [186] ... 42

FUSECO PG Future Seamless Communication Playground[30] ... 12

G-Lab German Lab[31] [187] ... 2

GEi Generic Enabler implementation ... 32

GENI Global Environment for Network Innovations[32] [13], [188] ... 2

GE Generic Enabler ... 32

GLOBECOM Global Communications Conference ... 173

GPS Global Positioning System ... 48

GUI Graphical User Interface ... 5

HADES Hades Active Delay Evaluation System ... 48

HPC High Performance Computing ... 14

HTML Hyper Text Markup Language ... 127

HTTP Hyper Text Transfer Protocol ... 69

IaaS Infrastructure as a Service ... 21

ICT Information and Communication Technology ... 2

ID Identifier ... 50

IEEE Institute of Electrical and Electronics Engineers ... 41

IETF Internet Engineering Task Force ... 53

INDL Infrastructure and Network Description Language [134] ... 60

INFINITY INfrastructures for the Future INternet CommunITY[33] ... 33

ICAF Intercloud Architecture Framework ... 40

ICMP Internet Control Message Protocol ... 5

IoC Internet of Content ... XXXVI

IoS Internet of Services ... 31

IoT Internet of Things [189] ... 2

IPFIX Internet Protocol Flow Information Export [120] ... 53

IPv6 Internet Protocol version 6 ... 112

[30] http://www.fuseco-playground.org

[31] http://german-lab.de

[32] http://geni.net

[33] http://fi-infinity.eu

IPTV Internet Protocol Television . . . 34
IP Internet Protocol . . . 54
ITG Informationstechnische Gesellschaft im VDE . . . 172
ITU-T International Telecommunication Union Telecommunication Standardization Sector . . . 40
IT Information Technology . . . 2
JAR Java Archive . . . 129
JCA-Cloud Joint Coordination Activity on Cloud Computing . . . 40
jFed jFed[34] . . . 130
JSON JavaScript Object Notation . . . 79
JSON-RPC JavaScript Object Notation - Remote Procedure Call . . . 5
KVM Kernel-based Virtual Machine . . . 24
LAMP Leveraging and Abstracting Measurements with PerfSONAR[35] . . . 49
LXC Linux Containers[36] . . . 26
M2M Machine-To-Machine Communication . . . 1
MaaS Monitoring as a Service . . . 47
MAFIA Monitoring Architecture for Federated heterogeneous Infrastructures 12
MCN Mobile Cloud Networking[37] [190] . . . 174
MIB Management Information Base . . . 54
MINER Measurement Infrastructure for Network Research [191] . . . 53
MOFI Monitoring Ontology for Federated Infrastructures [109], [159] . . . 100
MOMENT Monitoring and Measurement in the Next generation Technologies [34] 60
MP Measurement Point . . . 44
MySlice MySlice[38] . . . 48
NAT Network Address Translation . . . 113
NASA National Aeronautics and Space Administration . . . 107
NEPI Network Experimentation Programming Interface [192] . . . 42
NFV Network Function Virtualization [193] . . . 2

[34] http://jfed.iminds.be
[35] http://groups.geni.net/geni/wiki/LAMP
[36] https://linuxcontainers.org
[37] https://www.mobile-cloud-networking.eu
[38] http://myslice.info

NGN Next-Generation Network ... 29

NGSI Next-Generation Service Interface [194] ... 144

NIST National Institute of Standards and Technology [50] ... 19

NM-WG OGF Network Measurement Working Group ... 53

NML Network Mark-Up Language [195] ... 59

nmVO Network Measurement Virtual Observatory[39] [196] ... 48

NoF Network of the Future ... XXXVI

NOVI Networking innovations Over Virtualized Infrastructures [35] ... 48

NSF National Science Foundation ... 29

OASIS Organization for the Advancement of Structured Information Standards ... 18

OCCI Open Cloud Computing Interface [197] ... 26

OFELIA OpenFlow in Europe: Linking Infrastructure and Applications[40] ... 30

OGF Open Grid Forum ... 26

OMA Open Mobile Alliance ... 144

OMF cOntrol and Management Framework [198] ... 42

OML ORBIT Measurement Library [199] ... 42

Omni Omni[41] ... 130

OMN Open-Multinet[42] ... 102

OMSP OML Measurement Stream Protocol[43] ... 50

OneLab2 OneLab2 [200] ... 30

OneLab OneLab[44] [200] ... 30

OWAMP One-Way Active Measurement Protocol [201] ... 49

OOPS OntOlogy Pitfall Scanner[45] ... 152

OpenEPC Open Evolved Packet Core[46] [202] ... 143

OpenLab OpenLab[47] [203] ... 30

[39] http://nm.vo.elte.hu
[40] http://www.fp7-ofelia.eu
[41] http://trac.gpolab.bbn.com/gcf/wiki/Omni
[42] http://open-multinet.info
[43] http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html
[44] http://onelab.eu
[45] http://www.oeg-upm.net/oops
[46] http://www.openepc.net
[47] http://www.ict-openlab.eu/project-info.html

OPEX Operational Expenditure . . . 20

ORBIT Open Access Research Testbed for Next-Generation Wireless Networks [204] . . . 29

ORCA Open Resource Control Architecture [205] . . . 29

OS Operating System . . . 23

OVF Open Virtualization Format [206] . . . 27

OWL Web Ontology Language [126] . . . 57

OWL2 OWL 2 Web Ontology Language[48] . . . 152

PaaS Platform as a Service . . . 21

PerfSONAR Performance focused Service Oriented Network monitoring ARchitecture [38], [92] . . . 48

PHP PHP: Hypertext Preprocessor . . . 114

Panlab Pan European Laboratory Infrastructure Implementation [207] . . . 35

PlanetLab PlanetLab [208], [209][49] . . . 2

PM Physical Machine . . . 114

ProtoGENI ProtoGENI [210] . . . 29

Packet Tracking Multi-Hop Packet Tracking [211] . . . 48

QoS Quality of Service . . . 2

RAM Random-Access Memory . . . 147

RAG Red, Amber, and Green . . . 65

RDFS Resource Description Framework Schema [212] . . . 58

RDF Resource Description Framework [213] . . . 58

REST Representational State Transfer . . . 114

RSpec Resource Specification[50] . . . 129

SaaS Software as a Service . . . 21

SAWSDL Semantic Annotations for WSDL and XML Schema [214] . . . 59

SCP Secure Copy . . . 166

SCTP Stream Control Transmission Protocol . . . 55

SDN Software Defined Networking [215] . . . 2

SDO Standards Developing Organization . . . 26

[48] http://www.w3.org/TR/owl-overview/

[49] http://planet-lab.org

[50] http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs

SE Specific Enabler ... 33

Semantic Web Semantic Web [216] ... 56

SFA Slice-based Federation Architecture [217] ... 42

SIIF Standard for Intercloud Interoperability and Federation [47] ... 41

SLA Service Level Agreement ... 4

SmartSantander SmartSantander[51] ... 30

SME Small and Medium Enterprise ... 3

SMS Short Message Service ... 51

SNIA Storage Networking Industry Association ... 27

SNMP Simple Network Management Protocol [218] ... 48

SOA Service Oriented Architectures ... 18

SONoMA Service Oriented Network Measurement Architecture [219] ... 48

SPARQL SPARQL Protocol And RDF Query Language [220] ... 124

SQL Structured Query Language ... 51

SSH Secure Shell ... 166

SSN Semantic Sensor Network [136] ... 60

TCP Transmission Control Protocol ... 54

TDMI TopHat Dedicated Measurement Infrastructure ... 48

Teagle Teagle [8], [221] ... 42

Team Cymru Team Cymru[52] ... 48

TOSCA Topology and Orchestration Specification for Cloud Applications [222] 27

UCRE User-Customized Resource Environment ... 77

UDP User Datagram Protocol ... 54

URI Uniform Resource Identifier ... 95

URN Uniform Resource Name ... V

US United States ... 2

VANN Vocabulary for Annotating Vocabulary Descriptions ... 176

VCTTool Virtual Customer Testbed Tool [221] ... 129

VCT Virtual Customer Testbed ... 141

[51] http://smartsantander.eu

[52] http://www.team-cymru.org

VM Virtual Machine . . . 24
VMM Virtual Machine Manager . . . 24
VN Virtual Network . . . 45
VNF Virtualized Network Function . . . 63
VoIP Voice over Internet Protocol . . . 34
VPN Virtual Private Network . . . 78
WAN Wide Area Network . . . 78
W3C World Wide Web Consortium . . . 102
WMN Wireless Mesh Network . . . 45
WSDL Web Services Description Language . . . 59
XaaS Anything-as-a-Service . . . 19
XIFI Experimental Infrastructures for the Future Internet[53] [70] . . . 32
XML Extensible Markup Language . . . 53
XML-RPC Extensible Markup Language - Remote Procedure Call . . . 166
XSD XML Schema Definition . . . 102

[53] http://fi-xifi.eu

Bibliography

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper," 2014, [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html (cit. on pp. 1, 28).

[2] E. mobility report, "On the Pulse of the Networked Society," 2014, [Online]. Available: http://www.ericsson.com/res/docs/2014/ericsson-mobility-report-november-2014.pdf (cit. on pp. 1, 28).

[3] "ITU ICT Facts and Figures – The world in 2015," 2015, [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf (cit. on pp. 1, 28).

[4] T. Zahariadis, D. Papadimitriou, H. Tschofenig, S. Haller, P. Daras, G. D. Stamoulis, and M. Hauswirth, "Towards a Future Internet Architecture," in *Future Internet Assembly*, Springer LNCS, 2011, pp. 7–18, ISBN: 978-3-642-20897-3. DOI: 10.1007/978-3-642-20898-0_1. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20898-0_1 (cit. on pp. 1, 28, XXXVI).

[5] E. F. Group, "Fundamental Limitations of Current Internet and the path to Future Internet," 2011, [Online]. Available: http://www.future-internet.eu/publications (cit. on pp. 1, 28, XXXVI).

[6] J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 26–36, Jul. 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5936152. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2011.5936152 (cit. on pp. 2, 28).

[7] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," *Office*, vol. 121990, no. 1, p. 1, 1990. DOI: 10.1109/IEEESTD.1990.101064. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.1990.101064 (cit. on pp. 2, XXXVII).

[8] S. Wahle, "Generic Framework for Heterogeneous Resource Federation.," *Doctoral Dissertation.*, 2011. [Online]. Available: HTTP://opus4.kobv.de/opus4-tuberlin/files/3152/wahle_sebastian.pdf (cit. on pp. 2, 42, 183).

[9] S. Wahle, B. Harjoc, K. Campowsky, T. Magedanz, and A. Gavras, "Pan-European Testbed and Experimental Facility Federation - Architecture Refinement and Implementation," *International Journal of Communication Networks and Distributed Systems*, vol. 5, no. 1/2, p. 67, 2010, ISSN: 1754-3916. DOI: 10.1504/IJCNDS.2010.033968. [Online]. Available: http://dx.doi.org/10.1504/IJCNDS.2010.033968 (cit. on pp. 3, XXXVI).

[10] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, A. Quereilhac, Y. Al-Hazmi, F. Lobillo, C. Velayos, A. Vico-oton, and G. Androulidakis, "Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities," in *Future Network and Mobile Summit (FNMS)*, Lisboa, Portugal: IEEE, 2013, pp. 1–11, ISBN: 978-1-905824-37-3. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6633558 (cit. on pp. 3, 9, 32, 34, 35, 37, 42, 63–65, 83, 86).

[11] T. Wauters, B. Vermeulen, W. Vandenberghe, P. Demeester, S. Taylor, L. Baron, M. Smirnov, Y. Al-Hazmi, A. Willner, M. Sawyer, D. Margery, T. Rakotoarivelo, F. L. Vilela, D. Stavropoulos, C. Papagianni, F. Francois, C. Bermudo, A. Gavras, D. Davies, J. Lanza, and S.-Y. Park, "Federation of Internet Experimentation Facilities: Architecture and Implementation," in *European Conference on Networks and Communications*, IEEE, Jun. 2014, pp. 1–5. [Online]. Available: http://hdl.handle.net/1854/LU-5732987 (cit. on pp. 3, 7, 8, 30, 42, 46, 49, 64, 70, 72, 86, 178).

[12] A. Sanchez, I. Moerman, S. Bouckaert, D. Willkomm, J. Hauer, N. Michailow, G. Fettweis, L. Dasilva, J. Tallon, and S. Pollin, "Testbed federation: An approach for experimentation-driven research in cognitive radios and cognitive networking," *2011 Future Network & Mobile Summit*, pp. 1–9, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6095242 (cit. on pp. 3, 34, 46).

[13] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, Mar. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.12.037. [Online]. Available:

http://dx.doi.org/10.1016/j.bjp.2013.12.037 (cit. on pp. 2, 3, 29, 34, 46, 179).

[14] R. Ricci, G. Wong, L. Stoller, and J. Duerig, "An architecture for international federation of network testbeds," *IEICE Transactions on Communications*, vol. E96-B, no. 1, pp. 2–9, 2013. DOI: 10.1587/transcom.E96.B.2. [Online]. Available: http://doi.org/10.1587/transcom.E96.B.2 (cit. on pp. 3, 9).

[15] N. M. Calcavecchia, A. Celesti, and E. Di Nitto, "Understanding Decentralized and Dynamic Brokerage in Federated Cloud Environments," in *Achieving Federated and Self-Manageable Cloud Infrastructures*, IGI Global, 2012, pp. 36–56. DOI: 10.4018/978-1-4666-1631-8.ch003. [Online]. Available: http://dx.doi.org/10.4018/978-1-4666-1631-8.ch003 (cit. on p. 3).

[16] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Three-Phase Cross-Cloud Federation Model: The Cloud SSO Authentication," in *2010 Second International Conference on Advances in Future Internet*, IEEE, Jul. 2010, pp. 94–101, ISBN: 978-1-4244-7528-5. DOI: 10.1109/AFIN.2010.23. [Online]. Available: http://dx.doi.org/10.1109/AFIN.2010.23 (cit. on pp. 3, 40).

[17] A. C. Hume, Y. Al-Hazmi, B. Belter, K. Campowsky, L. M. Carril, G. Carrozzo, V. Engen, D. García-Pérez, J. Jofre Ponsatí, R. Kűbert, Y. Liang, C. Rohr, and G. Van Seghbroeck, "BonFIRE: A Multi-cloud Test Facility for Internet of Services Experimentation," in *8th International ICST Conference, TridentCom 2012*, vol. 44 LNICST, 2012, pp. 81–96. DOI: 10.1007/978-3-642-35576-9_11. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35576-9_11 (cit. on pp. 3, 7, 30, 31, 34, 49, 63, 64, 67, 68, 71, 72, 138, 177).

[18] D. Bernstein and Y. Demchenko, "The IEEE Intercloud Testbed – Creating the Global Cloud of Clouds," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, IEEE, Dec. 2013, pp. 45–50, ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.102. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.102 (cit. on pp. 3, 41).

[19] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, and Y. Al-Hazmi, "FIWARE Lab: Managing Resources and Services in a Cloud Federation Supporting Future Internet Applications," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, London, UK: IEEE, Dec. 2014, pp. 792–799, ISBN: 978-1-4799-7881-6. DOI: 10.1109/UCC.2014.129. [Online]. Available: http://dx.doi.org/10.1109/UCC.2014.129 (cit. on pp. 3, 8, 33, 46, 63, 64, 143, 144, 178).

[20] S. Paul, J. Pan, and R. Jain, "Architectures for the future networks and the next generation Internet: A survey," *Computer Communications*, vol. 34, no. 1, pp. 2–42, 2011. DOI: 10.1016/j.comcom.2010.08.001. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.08.001 (cit. on p. 3).

[21] A. Celesti, F. Tusa, and M. Villari, "Toward Cloud Federation," in *Achieving Federated and Self-Manageable Cloud Infrastructures*, IGI Global, 2012, pp. 1–17. DOI: 10.4018/978-1-4666-1631-8.ch001. [Online]. Available: http://dx.doi.org/10.4018/978-1-4666-1631-8.ch001 (cit. on pp. 3, 40, 72).

[22] S. Keranidis, D. Giatsios, T. Korakis, I. Koutsopoulos, L. Tassiulas, T. Rakotoarivelo, and T. Parmentelat, "Experimentation in Heterogeneous European Testbeds through the Onelab Facility: The Case of PlanetLab Federation with the Wireless NITOS Testbed," in *TRIDENTCOM2012*, Springer, 2012, pp. 338–354. DOI: 10.1007/978-3-642-35576-9_27. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35576-9_27 (cit. on pp. 3, 72).

[23] T. DeMarco, *Controlling Software Projects - Management, Measurement & Estimates*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1986, ISBN: 0131717111. [Online]. Available: http://dl.acm.org/citation.cfm?id=1096472 (cit. on p. 3).

[24] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *Proceedings of the International Conference on Parallel Processing Workshops*, 2010, pp. 275–279. DOI: 10.1109/ICPPW.2010.45. [Online]. Available: http://dx.doi.org/10.1109/ICPPW.2010.45 (cit. on p. 4).

[25] OGC, "Information Technology Infrastructure Library (ITIL) V3 Glossary of Terms and Definitions, Office of Government Commerce (now UK Cabinet Office)," 2007, [Online]. Available: http://www.best-management-practice.com/gempdf/itil_glossary_v3_1_24.pdf (cit. on p. 4).

[26] J. Ward and A. Barker, "Observing the clouds: a survey and taxonomy of cloud monitoring," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 3, no. 1, p. 24, 2014, ISSN: 2192-113X. DOI: 10.1186/s13677-014-0024-2. [Online]. Available: http://www.journalofcloudcomputing.com/content/3/1/24 (cit. on pp. 5, 46, 63, 68–70, 72).

[27] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, *A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives*, 2014. DOI: 10.1016/j.jpdc.2014.06.007. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2014.06.007 (cit. on pp. 5, 46, 63, 68–71).

[28] F. Alvarez, J. Gonzalez, F. M. Facca, and S. Cretti, "Technical and functional solutions to build a community cloud for future Internet services from an Infrastructure Owner Perspective," in *Workshop on Test beds for the Networks & Communications community (EUCNC)*, 2014 (cit. on pp. 7, 8, 49, 72).

[29] P. Calyam, C. Dovrolis, L. Jörgenson, R. Kettimuthu, B. Tierney, and J. Zurawski, "Monitoring and troubleshooting multi-domain networks using measurement federations [Guest Editorial]," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 53–54, Nov. 2013, ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6658652. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2013.6658652 (cit. on pp. 7, 45, 49, 72).

[30] "Elsevier Future Generation Computer Systems Special Issue on "Cloud Monitoring Systems"," 2012, [Online]. Available: http://www.elsevierscitech.com/lmfile/otherformat/1522_Future.pdf (cit. on pp. 7, 49, 72, 77).

[31] G. Aceto, A. Botta, W. de Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013, ISSN: 13891286. DOI: 10.1016/j.comnet.2013.04.001. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2013.04.001 (cit. on pp. 7, 46, 49, 69, 70, 72).

[32] C. Zeginis, K. Kritikos, P. Garefalakis, K. Konsolaki, K. Magoutis, and D. Plexousakis, "Towards Cross-Layer Monitoring of Multi-Cloud Service-Based Applications," in *The Second European Conference on Service-Oriented and Cloud Computing (ESOCC)*, vol. 8135, Springer LNCS, 2013, pp. 188–195. DOI: 10.1007/978-3-642-40651-5_16. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40651-5_16 (cit. on pp. 7, 8, 45).

[33] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, vol. 22, no. 3, pp. 183–236, 1990, ISSN: 03600300. DOI: 10.1145/96602.96604. [Online]. Available: http://dx.doi.org/10.1145/96602.96604 (cit. on p. 7).

[34] A. Salvador, J. E. L. de Vergara, G. Tropea, N. Blefari-Melazzi, Á. Ferreiro, and Á. Katsu, "A Semantically Distributed Approach to Map IP Traffic Measurements to a Standardized Ontology," *Int. Journal of Computer Networks & Communications*, vol. 2, no. 1, pp. 13–31, 2010. DOI: 10.1.1.158.9560. [Online]. Available: http://airccse.org/journal/cnc/0110s02.pdf (cit. on pp. 8, 60, 107, 180).

[35] J. van der Ham, J. Stéger, S. Laki, Y. Kryftis, V. Maglaris, and C. de Laat, "The NOVI information models," *Future Generation Computer Systems*, vol. 42, pp. 64–73, 2015, ISSN: 0167-739X. DOI: 10.1016/j.future.2013.12.017. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.12.017 (cit. on pp. 8, 48, 60, 181).

[36] ETSI-MOI-Group Specification, "European Telecommunications Standards Institute (ETSI) Industry Group MOI (Measurement Ontology for IP traffic)," 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/moi/001_099/003/01.01.01_60/gs_moi003v010101p.pdf (cit. on pp. 8, 60, 103).

[37] E. Boschi, S. D'Antonio, P. Malone, and C. Schmoll, "INTERMON: An Architecture for Inter-domain Monitoring, Modelling and Simulation," in *4th International IFIP-TC6 Networking Conference*, 2005, pp. 1397–1400. DOI: 10.1007/11422778_123. [Online]. Available: http://dx.doi.org/10.1007/11422778_123 (cit. on pp. 8, 45).

[38] B. L. Tierney, J. Boote, E. Boyd, A. Brown, M. Grigoriev, J. Metzger, M. Swany, M. Zekauskas, and J. Zurawski, "Instantiating a Global Network Measurement Framework," Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, Tech. Rep., Dec. 2008. DOI: 10.2172/946807. [Online]. Available: http://dx.doi.org/10.2172/946807 (cit. on pp. 8, 49, 182).

[39] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 7.1–7.47, 2014. DOI: 10.1145/2593512. [Online]. Available: http://dx.doi.org/10.1145/2593512 (cit. on pp. 9, 10, 35, 69).

[40] H. H. Aberra, "What is SAP Business Blueprint?" In *Handbook of Research on Enterprise Systems*, IGI Global, 2009, pp. 19–31. DOI: 10.4018/978-1-59904-859-8.ch002. [Online]. Available: http://dx.doi.org/10.4018/978-1-59904-859-8.ch002 (cit. on p. 18).

[41] OASIS, "Reference Model for Service Oriented Architecture 1.0," *Organization for the Advancement of Structured Information Standards (OASIS) Std. 1.0*, 2006. [Online]. Available: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf (cit. on p. 18).

[42] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-Oriented Cloud Computing Architecture," in *2010 Seventh International Conference on Information Technology: New Generations*, IEEE, 2010, pp. 684–689, ISBN: 978-1-4244-6270-4. DOI: 10.1109/ITNG.2010.214. [Online]. Available: http://dx.doi.org/10.1109/ITNG.2010.214 (cit. on p. 18).

[43] G. Feuerlicht, "Next Generation SOA: Can SOA Survive Cloud Computing?" In *Advances in Intelligent and Soft Computing*, vol. 67 AISC, 2010, pp. 19–29, ISBN: 978-3-642-10686-6. DOI: 10.1007/978-3-642-10687-3_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10687-3_2 (cit. on p. 18).

[44] G. Raines, "Cloud Computing and SOA," *Systems Engineering at MITRE, Service-oriented architecture (SOA) series*, 2009. [Online]. Available: http://www.mitre.org/sites/default/files/pdf/09_0743.pdf (cit. on p. 18).

[45] T. Magedanz, F. Schreiner, and S. Wahle, "Service-oriented testbed infrastructures and cross-domain federation for future internet research," in *2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops, IM 2009*, 2009, pp. 101–106. DOI: 10.1109/INMW.2009.5195944. [Online]. Available: http://dx.doi.org/10.1109/INMW.2009.5195944 (cit. on pp. 19, 34).

[46] N. Blum, T. Magedanz, F. Schreiner, and S. Wahle, "Service Oriented Testbed Infrastructures: a Cross-Layer Approach for NGNs," *Mobile Networks and Applications*, vol. 15, no. 3, pp. 413–424, Jun. 2010, ISSN: 1383-469X. DOI: 10.1007/s11036-009-0201-6. [Online]. Available: http://dx.doi.org/10.1007/s11036-009-0201-6 (cit. on p. 19).

[47] D. Bernstein and V. Deepak, "Draft Standard for Intercloud Interoperability and Federation (SIIF)," IEEE P2303, Tech. Rep., 2014. [Online]. Available: https://standards.ieee.org/develop/project/2302.html (cit. on pp. 19, 34, 41, 183).

[48] A. Shawish and M. Salama, "Cloud Computing: Paradigms and Technologies," in *Studies in Computational Intelligence*, vol. 495, 2014, pp. 39–67. DOI: 10.1007/978-3-642-35016-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35016-0_2 (cit. on p. 20).

[49] R.-A. C. Nguemaleu and L. Montheu, *Roadmap to Greener Computing*. Chapman and Hall/CRC, pp. 1–246, ISBN: 9781466506848 (cit. on p. 21).

[50] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST, Tech. Rep., 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (cit. on pp. 19, 22, 181).

[51] S. Nanda and T. Chiueh, "A Survey on Virtualization Technologies," *Science*, vol. 179, no. Vm, U. A. S. Brook, Ed., pp. 1–42, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.371 (cit. on pp. 23, 24).

[52] J. Carapinha and J. Jiménez, "Network virtualization: a view from the bottom," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures - VISA '09*, ser. VISA '09, New York, New York, USA: ACM Press, 2009, p. 73, ISBN: 9781605585956. DOI: 10.1145/1592648.1592660. [Online]. Available: http://dx.doi.org/10.1145/1592648.1592660 (cit. on p. 23).

[53] A. Berl, A. Fischer, and H. de Meer, "Using System Virtualization to Create Virtualized Networks," *Electronic Communications of the EASST*, vol. 17, pp. 1–12, 2009. [Online]. Available: http://www.net.fim.uni-passau.de/pdf/Berl2009e.pdf (cit. on p. 23).

[54] A. Galis, S. Clayman, A. Fischer, A. Paler, Y. Al-Hazmi, H. De Meer, A. Cheniour, O. Mornard, J. P. Gelas, L. Lefevre, J. R. Loyola, A. Astorga, J. Serrat, and S. Davy, "Future Internet Management Platforms for Network Virtualisation and Service Clouds," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6481 LNCS, 2010, pp. 235–237. DOI: 10.1007/978-3-642-17694-4_39. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17694-4_39 (cit. on p. 23).

[55] S. Sargento, R. Matos, K. A. Hummel, A. Hess, S. Toumpis, Y. Tselekounis, G. D. Stamoulis, Y. Al-Hazmi, M. Ali, and H. de Meer, "Multi-Access Communications in Wireless Mesh Networks by Virtualization," in *Wireless Multi-Access Environments and Quality of Service Provisioning*, IGI Global, 2011, pp. 97–138, ISBN: 9781466600171. DOI: 10.4018/978-1-4666-0017-1.ch005. [Online]. Available: http://dx.doi.org/10.4018/978-1-4666-0017-1.ch005 (cit. on pp. 23, 45, 63).

[56] Y. Al-Hazmi and H. de Meer, "Virtualization of 802.11 Interfaces for Wireless Mesh Networks," in *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, IEEE, Jan. 2011, pp. 44–51, ISBN: 978-1-61284-189-2. DOI: 10.1109/WONS.2011.5720199. [Online]. Available: http://dx.doi.org/10.1109/WONS.2011.5720199 (cit. on pp. 23, 45).

[57] I. Coskun, Hakan Schieferdecker and Y. Al-Hazmi, "Virtual WLAN: Going beyond Virtual Access Points," *Electronic Communications of the EASST*, vol. 25, no. 2, pp. 32–38, 2009, ISSN: 1863-2122. [Online]. Available: http://journal.ub.tu-berlin.de/eceasst/article/view/226 (cit. on p. 23).

[58] H. Basilier, M. Darula, and J. Wilke, "Virtualizing network services - the telecom cloud," *Ericsson Review, The communications Technology Journal*, pp. 1–9, 2014, ISSN: 0014-0171. [Online]. Available: http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2014/er-telecom-cloud.pdf (cit. on p. 23).

[59] M. Pathirage, S. Perera, I. Kumara, and S. Weerawarana, "A Multi-tenant Architecture for Business Process Executions," in *2011 IEEE International Conference on Web Services*, IEEE, Jul. 2011, pp. 121–128, ISBN: 978-1-4577-0842-8. DOI: 10.1109/ICWS.2011.99. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2011.99 (cit. on p. 26).

[60] R. Krebs, C. Momm, and S. Kounev, "Architectural Concerns in Multi-tenant SaaS Applications," *CLOSER*, pp. 426–431, 2012. [Online]. Available: https://sdqweb.ipd.kit.edu/publications/pdfs/KrMoKo2012-closer-multitenant-sass.pdf (cit. on p. 26).

[61] S. Strauch, V. Andrikopoulos, S. Gómez Sáez, and F. Leymann, "ESBˆMT: A Multi-tenant Aware Enterprise Service Bus," *International Journal of Next-Generation Computing*, vol. 4, no. 3, pp. 230–249, 2013 (cit. on p. 26).

[62] G. Katsaros, M. Menzel, A. Lenk, J. Rake-Revelant, R. Skipp, and J. Eberhardt, "Cloud Application Portability with TOSCA, Chef and Openstack," in *2014 IEEE International Conference on Cloud Engineering (IC2E'14)*, 2014, pp. 295–302. DOI: 10.1109/IC2E.2014.27. [Online]. Available: http://dx.doi.org/10.1109/IC2E.2014.27 (cit. on p. 27).

[63] A. Ciuffoletti, "A Simple and Generic Interface for a Cloud Monitoring Service," in *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, SCITEPRESS - Science, 2014, pp. 143–150, ISBN: 978-989-758-019-2. DOI: 10.5220/0004940901430150. [Online]. Available: http://dx.doi.org/10.5220/0004940901430150 (cit. on p. 27).

[64] Y. Al-Hazmi, K. Campowsky, and T. Magedanz, "A Monitoring System for Federated Clouds," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, IEEE, Nov. 2012, pp. 68–74, ISBN: 978-1-4673-2798-5. DOI: 10.1109/CloudNet.2012.6483657. [Online]. Available: http://dx.doi.org/10.1109/CloudNet.2012.6483657 (cit. on pp. 27, 40, 45, 63, 64, 69, 78, 82, 112, 115, 123, 136, 138, 139).

[65] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future Internet Research and Experimentation: The FIRE Initiative," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 89–92, 2007, ISSN: 0146-4833. DOI: 10.1145/1273445.1273460. [Online]. Available: http://doi.acm.org/10.1145/1273445.1273460 (cit. on pp. 2, 28, 30, 178, XXXVI).

[66] P. Stuckmann and R. Zimmermann, "European Research on Future Internet Design"," *IEEE Wireless Communications Magazine*, vol. 16, no. 5, pp. 14–22, 2009. DOI: 10.1109/MWC.2009.5300298. [Online]. Available: http://dx.doi.org/10.1109/MWC.2009.5300298 (cit. on pp. 28, XXXVI).

[67] T. Magedanz and S. Wahle, "Control framework design for Future Internet testbeds," *e & i Elektrotechnik und Informationstechnik*, vol. 126, pp. 274–279, 2009, ISSN: 0932-383X. DOI: 10.1007/s00502-009-0655-z. [Online]. Available: http://dx.doi.org/10.1007/s00502-009-0655-z (cit. on p. 29).

[68] D. Kim, J. Kim, G. Wang, J.-H. Park, and S.-H. Kim, "K-GENI testbed deployment and federated meta operations experiment over GENI and KREONET," *Computer Networks*, vol. 61, pp. 39–50, Mar. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.11.016. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.11.016 (cit. on p. 29).

[69] A. Quereilhac, A. Willner, Y. Al-Hazmi, C. Tranoris, P. Becue, V. Sercu, J. Auge, T. Rakotoarivelo, A. Gulyas, G. Biczok, and P. Chrysa, "OpenLab: Experimental plane – Experiment Controllers," European FP7 Project OpenLabs, Tech. Rep., 2012, pp. 1–46. [Online]. Available: http://www.ict-openlab.eu/fileadmin/documents/public_deliverables/OpenLab_Deliverable_D2_1.pdf (cit. on p. 31).

[70] Y. Al-Hazmi, A. Willner, B. Pickering, A. Alloush, T. Magedanz, and S. Cretti, "Creating a Sustainable Federation of Cloud-Based Infrastructures for the Future Internet — The FIWARE Approach," in *10th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2015)*, Vancouver, Canada: ACM, 2015, pp. 1–10. DOI: 10.4108/icst.tridentcom.2015.259747. [Online]. Available: http://dx.doi.org/10.4108/icst.tridentcom.2015.259747 (cit. on pp. 33–36, 38, 41, 65, 66, 86, 143, 144, 184).

[71] M. Serrano, S. Davy, M. Johnsson, W. Donnelly, and A. Galis, "Review and Designs of Federated Management in Future Internet Architectures," in *International Journal of Network Management*, 6, vol. 22, 2011, pp. 51–66. DOI: 10.1007/978-3-642-20898-0_4. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20898-0_4 (cit. on p. 34).

[72] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation," in *2010 IEEE 3rd International Conference on Cloud Computing*, IEEE, Jul. 2010, pp. 337–345, ISBN: 978-1-4244-8207-8. DOI: 10.1109/CLOUD.2010.46. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2010.46 (cit. on p. 34).

[73] O. Appleton, "D3.1: Business models for Federated e-Infrastructures," FedSM, Tech. Rep., 2012. [Online]. Available: http://fitsm.itemo.org/sites/default/files/FedSM-D3.1-Business_models-v1.0.pdf (cit. on pp. 34–36, 38, 178).

[74] M. Serrano, S. Davy, M. Johnsson, W. Donnelly, and A. Galis, "Review and Designs of Federated Management in Future Internet Architectures," in *The Future Internet - Future Internet Assembly 2011: Achievements and Technological Promises*, 2011, pp. 51–66. DOI: 10.1007/978-3-642-20898-0_4. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20898-0_4 (cit. on p. 34).

[75] A. Gavras, T. Magedanz, S. Wahle, H. Hrasnica, S. Avéssta, and J.-C. Imbeaux, "Deliverable D2.1: Legal Framework (Version 2)," Panlab, Tech. Rep., 2008, pp. 1–34. [Online]. Available: http://www.panlab.net/fileadmin/documents/Deliverables/Panlab-D2.1-Legal-Framework-V2.0.pdf (cit. on p. 35).

[76] J.-C. Imbeaux, A. Gavras, H. Hrasnica, S. Wahle, O. Martinot, and S. Avéssta, "Deliverable D2.1: Vision for a Pan-European Laboratory (Version 2)," Panlab, Tech. Rep., 2007, pp. 1–52. [Online]. Available: http://www.panlab.net/fileadmin/documents/Deliverables/Panlab-D1.2-Vision-for-PEL-V2.0.pdf (cit. on p. 35).

[77] J. Van Ooteghem, S. Taylor, P. Grace, F. Lobillo, M. Smirnov, and P. Demeester, "Sustaining a federation of Future Internet experimental facilities," *International Telecommunications Society (ITS) conference*, 2014. [Online]. Available: http://hdl.handle.net/10419/101436 (cit. on pp. 35, 36, 38, 43, 66, 70, 71).

[78] J. Van Ooteghem, B. Naudts, W. Vandenberghe, B. Vermeulen, S. Bouckaert, S. Taylor, P. Grace, F. Lobillo, J. Martrat, L. Muñoz, P. Sotres, and M. Sawyer, "Deliverable D2.3: First Sustainability Plan," Fed4FIRE, Tech. Rep., 2013. [Online]. Available: http://www.fed4fire.eu/fileadmin/documents/public_deliverables/D2-3_Fed4FIRE_First_sustainability_plan.pdf (cit. on p. 38).

[79] M. Heikkurinen, O. Appleton, L. Urciuoli, and J. Hintsa, "Federated ICT for global supply chains: IT service management in cross-border trade," in *2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*, ser. 2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013, 2013, pp. 1268–1275, ISBN: 978-1-4673-5229-1. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6573176 (cit. on p. 39).

[80] D. García-Pérez, J. Á. Lorenzo del Castillo, Y. Al-Hazmi, J. Martrat, K. Kavoussanakis, A. C. Hume, C. V. López, G. Landi, T. Wauters, M. Gienger, and D. Margery, "Cloud and Network Facilities Federation in BonFIRE," in *Federative and interoperable cloud infrastructures, Aug 2013, Aachen, Germany. Euro-Par 2013: Parallel Processing Workshops, 8374, Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2014, pp. 126–135. DOI: 10.1007/978-3-642-54420-0_13. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54420-0_13 (cit. on pp. 40, 41, 136, 138).

[81] B. Rochwerger, C. Vázquez, D. Breitgand, D. Hadas, M. Villari, P. Massonet, E. Levy, A. Galis, I. M. Llorente, R. S. Montero, Y. Wolfsthal, K. Nagin, L. Larsson, and F. Galán, "An Architecture for Federated Cloud Computing," in *Cloud Computing*, Hoboken, NJ, USA: John Wiley & Sons, Inc., Jan.

2011, pp. 391–411. DOI: 10.1002/9780470940105.ch15. [Online]. Available: http://doi.wiley.com/10.1002/9780470940105.ch15 (cit. on p. 40).

[82] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti, "Cloud Federations in Contrail," in *Euro-Par 2011: Parallel Processing Workshops*, Springer, 2012, pp. 159–168. DOI: 10.1007/978-3-642-29737-3_19. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29737-3_19 (cit. on p. 40).

[83] Y. Demchenko, C. Ngo, C. de Laat, J. Rodriguez, L. M. Contreras, J. A. Garcia-Espin, S. Figuerola, G. Landi, and N. Ciulli, "Intercloud Architecture Framework for Heterogeneous Cloud Based Infrastructure Services Provisioning On-Demand," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, IEEE, Mar. 2013, pp. 777–784, ISBN: 978-1-4673-6239-9. DOI: 10.1109/WAINA.2013.237. [Online]. Available: http://dx.doi.org/10.1109/WAINA.2013.237 (cit. on p. 40).

[84] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. Masoud Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, Sep. 2012, ISSN: 00220000. DOI: 10.1016/j.jcss.2011.12.017. [Online]. Available: http://dx.doi.org/10.1016/j.jcss.2011.12.017 (cit. on p. 41).

[85] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," *RFC 5280*, pp. 1–152, 2008. [Online]. Available: http://tools.ietf.org/html/rfc5280 (cit. on p. 41).

[86] J. A. S. Monteiro, "Measurement Infrastructures for Future Internet Testbeds," in *First Workshop of the Brazilian Institute for Web Science Research. Rio de Janeiro*, 2010, pp. 1–10. [Online]. Available: http://webscience.org.br/wiki/images/4/49/Fia.g3-20100625-suruagy.pdf (cit. on pp. 42, 47, 63, 67, 68, 71).

[87] J. Weiner, "Measurement: Reliability and Validity Measures," *Mimeo, Bloomberg School of Public Health, Johns Hopkins University*, 2007. [Online]. Available: http://ocw.jhsph.edu/courses/hsre/PDFs/HSRE_lect7_weiner.pdf (cit. on pp. 44, XXXVI).

[88] G. Aceto, A. Botta, W. de Donato, and A. Pescape, "Cloud monitoring: Definitions, issues and future directions," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, IEEE, Nov. 2012, pp. 63–67, ISBN: 978-1-4673-2798-5. DOI: 10.1109/CloudNet.2012.6483656. [Online]. Available: http://dx.doi.org/10.1109/CloudNet.2012.6483656 (cit. on pp. 45, 62, 63, 68, 69).

[89] A. Mandal, I. Baldin, Y. Xin, P. Ruth, and C. Heerman, "Enabling persistent queries for cross-aggregate performance monitoring," *Communications Magazine, IEEE*, vol. 52, no. 5, pp. 157–164, 2014. DOI: 10.1109/MCOM.2014.6815907. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2014.6815907 (cit. on p. 45).

[90] Y. Al-Hazmi, H. de Meer, K. A. Hummel, H. Meyer, M. Meo, and D. Remondo, "Energy-Efficient Wireless Mesh Infrastructures," *IEEE Network*, vol. 25, no. 2, pp. 32–38, Mar. 2011, ISSN: 0890-8044. DOI: 10.1109/MNET.2011.5730526. [Online]. Available: http://dx.doi.org/10.1109/MNET.2011.5730526 (cit. on p. 45).

[91] T. Baur and S. Saad, "Virtualizing Resources: Customer-Oriented Cross-Domain Monitoring for Service Grids," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, IEEE, May 2007, pp. 777–780, ISBN: 1-4244-0798-2. DOI: 10.1109/INM.2007.374711. [Online]. Available: http://dx.doi.org/10.1109/INM.2007.374711 (cit. on p. 45).

[92] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. M. Swany, S. Trocha, and J. Zurawski, "PerfSONAR: A Service Oriented Architecture for Multi-domain Network Monitoring," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3826 LNCS, 2005, pp. 241–254. DOI: 10.1007/11596141_19. [Online]. Available: http://dx.doi.org/10.1007/11596141_19 (cit. on pp. 45, 49, 182).

[93] Y. Xin, I. Baldin, J. Chase, K. Ogan, and K. Anyanwu, "Leveraging Semantic Web Technologies for Managing Resources in a Multi-Domain Infrastructure-as-a-Service Environment," pp. 1–20, 2014. arXiv: 1403.0949. [Online]. Available: http://arxiv.org/abs/1403.0949 (cit. on p. 45).

[94] Y. Al-Hazmi and T. Magedanz, "A Flexible Monitoring System for Federated Future Internet Testbeds," in *2012 Third International Conference on The Network of the Future (NOF)*, IEEE, Nov. 2012, pp. 1–6, ISBN: 978-1-4673-5265-9. DOI: 10.1109/NOF.2012.6463985. [Online]. Available: http://dx.doi.org/10.1109/NOF.2012.6463985 (cit. on pp. 45, 63, 64, 67–69, 71, 92, 128, 141, 145).

[95] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, C. Dobre, A. Muraru, A. Costan, M. Dediu, and C. Stratan, "MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems," *Computer Physics Communications*, vol. 180, no. 12, pp. 2472–2498, Dec. 2009, ISSN: 00104655. DOI: 10.1016/j.cpc.2009.08.003. [Online]. Available: http://dx.doi.org/10.1016/j.cpc.2009.08.003 (cit. on p. 46).

[96] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mOSAIC ontology for Cloud resources annotation," in *Federated Conf. on Computer Science and Information Systems (FedCSIS)*, IEEE, 2011, pp. 973–980. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6078209 (cit. on p. 46).

[97] V. C. Emeakaroha, T. C. Ferreto, M. A. S. Netto, I. Brandic, and C. A. F. De Rose, "CASViD: Application Level Monitoring for SLA Violation Detection in Clouds," in *2012 IEEE 36th Annual Computer Software and Applications Conference*, IEEE, Jul. 2012, pp. 499–508, ISBN: 978-1-4673-1990-4. DOI: 10.1109/COMPSAC.2012.68. [Online]. Available: http://dx.doi.org/10.1109/COMPSAC.2012.68 (cit. on p. 46).

[98] G. Katsaros, R. Kübert, and G. Gallizo, "Building a service-oriented monitoring framework with REST and nagios," in *Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011*, 2011, pp. 426–431. DOI: 10.1109/SCC.2011.53. [Online]. Available: http://dx.doi.org/10.1109/SCC.2011.53 (cit. on p. 46).

[99] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. Oriol Fitó, and D. Henriksson, "A multi-level architecture for collecting and managing monitoring information in cloud environments," in *1st Int. Conference on Cloud Computing and Services Science*, 2011, pp. 1–4. DOI: 10.5220/0003388602320239. [Online]. Available: http://dx.doi.org/10.5220/0003388602320239 (cit. on p. 46).

[100] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. Fitó, and D. Espling, "An Integrated Monitoring Infrastructure for Cloud Environments," in *Cloud Computing and Services Science, Service Science: Research and Innovations in the Service Economy 2012*, Springer New York, 2012, pp. 149–164. DOI: 10.1007/978-1-4614-2326-3_8. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-2326-3_8 (cit. on p. 46).

[101] J. Tordsson, K. Djemame, D. Espling, G. Katsaros, W. Ziegler, O. Wäldrich, K. Konstanteli, A. Sajjad, M. Rajarajan, G. Gallizo, and S. Nair, "Towards Holistic Cloud Management," in *European Research Activities in Cloud Computing*, Cambridge Scholars Publishing, 2012, pp. 122–150, ISBN: 978-1-4438-3507-7 (cit. on p. 46).

[102] S. Vinoski, "RESTful Web Services Development Checklist," *IEEE Internet Computing*, vol. 12, no. 6, pp. 96–95, Nov. 2008, ISSN: 1089-7801. DOI: 10.1109/MIC.2008.130. [Online]. Available: http://dx.doi.org/10.1109/MIC.2008.130 (cit. on p. 46).

[103] S. De Chaves, R. Uriarte, and C. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 130–137, Dec. 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.6094017. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2011.6094017 (cit. on p. 47).

[104] B. König, J. Alcaraz Calero, and J. Kirschnick, "Elastic monitoring framework for cloud infrastructures," *IET Communications*, vol. 6, no. 10, p. 1306, 2012, ISSN: 17518628. DOI: 10.1049/iet-com.2011.0200. [Online]. Available: http://dx.doi.org/10.1049/iet-com.2011.0200 (cit. on p. 47).

[105] S. Clayman, G. Toffetti, A. Galis, and C. Chapman, "Monitoring services in a federated cloud – the reservoir experience," in *Achieving Federated and Self-Manageable Cloud Infrastructures*, IGI Global, 2012. DOI: 10.4018/978-1-4666-1631-8.ch013. [Online]. Available: http://dx.doi.org/10.4018/978-1-4666-1631-8.ch013 (cit. on p. 47).

[106] J. Griffioen, Z. Fei, and H. Nasir, "Architectural Design and Specification of the INSTOOLS Measurement System," *Laboratory of Advanced Networking, University of Kentucky, Tech. Rep*, 2009 (cit. on pp. 47, 53).

[107] J. Griffioen, Z. Fei, H. Nasir, X. Wu, J. Reed, and C. Carpenter, "Measuring experiments in GENI," *Computer Networks*, vol. 63, pp. 17–32, Apr. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.10.016. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.10.016 (cit. on pp. 47, 53, 63, 64, 67, 68, 71).

[108] T. Bourgeau, J. Augé, and T. Friedman, "TopHat: Supporting Experiments through Measurement Infrastructure Federation," in *Testbeds and Research Infrastructures. Development of Networks and Communities*, Springer, 2011, pp. 542–557. DOI: 10.1007/978-3-642-17851-1_41. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17851-1_41 (cit. on pp. 48, 53).

[109] Y. Al-Hazmi and T. Magedanz, "Towards Semantic Monitoring Data Collection and Representation in Federated Infrastructures," in *2015 3rd International Conference on Future Internet of Things and Cloud*, IEEE, Aug. 2015, pp. 17–24, ISBN: 978-1-4673-8103-1. DOI: 10.1109/FiCloud.2015.40. [Online]. Available: http://dx.doi.org/10.1109/FiCloud.2015.40 (cit. on pp. 50, 56, 69, 100, 103, 109, 180).

[110] A. Pras and J. Schoenwaelder, "On the Difference between Information Models and Data Models," Internet Engineering Task Force, Tech. Rep., 2003, pp. 1–8. [Online]. Available: http://www.ietf.org/rfc/rfc3444.txt (cit. on p. 52).

[111] A. Y. Levy, "The Information Manifold Approach to Data Integration," *EEE Intelligent Agents*, vol. 13, no. 5, pp. 12–16, 1998. DOI: 10.1.1.50.1593. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.1593 (cit. on p. 52).

[112] M. Reddy, B. Prasad, P. Reddy, and A. Gupta, "A methodology for integration of heterogeneous databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 6, pp. 920–933, 1994, ISSN: 10414347. DOI: 10.1109/69.334882. [Online]. Available: http://dx.doi.org/10.1109/69.334882 (cit. on p. 52).

[113] M. N. Kamel and M. Zviran, "Heterogeneous databases integration in a hospital information systems environment: a bottom-up approach," *Proceedings / the ... Annual Symposium on Computer Application [sic] in Medical Care. Symposium on Computer Applications in Medical Care*, pp. 363–367, 1991. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/1807623 (cit. on p. 52).

[114] J. Zurawski, M. Swany, and D. Gunter, "A Scalable Framework for Representation and Exchange of Network Measurements," in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, vol. 2006, IEEE, 2006, pp. 409–417, ISBN: 1-4244-0106-2. DOI: 10.1109/TRIDNT.2006.1649176. [Online]. Available: http://dx.doi.org/10.1109/TRIDNT.2006.1649176 (cit. on p. 53).

[115] S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, N. Mendelsohn, D. Beech, and M. Maloney, *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*, 2012. [Online]. Available: http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/ (cit. on pp. 53, 102).

[116] J. E. L. de Vergara, J. Aracil, J. Martínez, A. Salvador, and J. A. Hernández, "Application of ontologies for the integration of network monitoring platforms," in *Proceedings of the 1st European Workshop on Mechanisms for Mastering Future Internet. Salzburg, Austria*, 2008. [Online]. Available: http://www.math.jhu.edu/jmartinezgarcia/papers/MonitoringOntology.pdf (cit. on p. 53).

[117] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," Tech. Rep., 2013, p. 76. [Online]. Available: https://tools.ietf.org/html/rfc7011 (cit. on pp. 53, 54).

[118] V. Bajpai and J. Schonwalder, "A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2015, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2418435. [Online]. Available: http://dx.doi.org/10.1109/COMST.2015.2418435 (cit. on p. 54).

[119] P. Goncalves, J. L. Oliveira, and R. L. Aguiar, "An evaluation of network management protocols," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*, IEEE, Jun. 2009, pp. 537–544, ISBN: 978-1-4244-3486-2. DOI: 10.1109/INM.2009.5188859. [Online]. Available: http://dx.doi.org/10.1109/INM.2009.5188859 (cit. on p. 54).

[120] E. Boschi, B. Trammell, L. Mark, and T. Zseby, "Exporting type information for IP flow information export (IPFIX) information elements," RFC 5610, RFC Editor, Fremont, CA, USA, Tech. Rep., 2009. [Online]. Available: https://tools.ietf.org/html/rfc5610 (cit. on pp. 53, 54, 179).

[121] T. Kothmayr, C. Schmitt, L. Braun, and G. Carle, "Gathering Sensor Data in Home Networks with IPFIX," in *Wireless Sensor Networks*, Springer, 2010, pp. 131–146. DOI: 10.1007/978-3-642-11917-0_9. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11917-0_9 (cit. on p. 55).

[122] NetQos, "White Paper: Best Practices for Cisco NetFlow/IPFIX Analysis and Reporting," 2008. [Online]. Available: http://www.techieindex.com/techie1/whitepapers/wp_details.jsp?id=4447 (cit. on p. 55).

[123] O. Mehani, G. Jourjon, T. Rakotoarivelo, and M. Ott, "An instrumentation framework for the critical task of measurement collection in the future Internet," *Computer Networks*, vol. 63, pp. 68–83, Apr. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2014.01.007. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2014.01.007 (cit. on pp. 55, 69, 70, 152, 155, 164).

[124] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.7493 (cit. on pp. 56, 101).

[125] B. Chandrasekaran, J. Josephson, and V. Benjamins, "What are ontologies, and why do we need them?" *IEEE Intelligent Systems*, vol. 14, no. 1, pp. 20–26, Jan. 1999, ISSN: 1094-7167. DOI: 10.1109/5254.747902. [Online]. Available: http://dx.doi.org/10.1109/5254.747902 (cit. on p. 56).

[126] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," W3C, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/owl-features/ (cit. on pp. 57, 182).

[127] M. Bergman, *Advantages and Myths of RDF*, 2009. [Online]. Available: http://www.mkbergman.com/483/advantages-and-myths-of-rdf/ (cit. on pp. 57, 100).

[128] A. Marchetti, F. Ronzano, M. Tesconi, and M. Minutoli, "Formalizing Knowledge by Ontologies: OWL and KIF," *Relatório apresentado L'Istituto di Informatica e Telematica (IIT). Consiglio Nazionale delle Ricerche (CNR). Italia*, 2008 (cit. on p. 57).

[129] A. Willner, C. Papagianni, M. Giatili, P. Grosso, M. Morsey, Y. Al-Hazmi, and I. Baldin, "Open-Multinet Upper Ontology — Towards the Semantic-based Management of Federated Infrastructures," in *10th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2015)*, Vancouver, Canada: ACM, 2015, pp. 1–10. DOI: 10.4108/icst.tridentcom.2015.259750. [Online]. Available: http://dx.doi.org/10.4108/icst.tridentcom.2015.259750 (cit. on pp. 59, 102).

[130] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things," *International Journal on Semantic Web and Information Systems*, vol. 8, no. 1, pp. 1–21, Jan. 2012, ISSN: 1552-6283. DOI: 10.4018/jswis.2012010101. [Online]. Available: http://dx.doi.org/10.4018/jswis.2012010101 (cit. on p. 59).

[131] J. M. Serrano Orozco, *Applied Ontology Engineering in Cloud Services, Networks and Management Systems*. Boston, MA: Springer US, 2012, ISBN: 978-1-4614-2235-8. DOI: 10.1007/978-1-4614-2236-5. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-2236-5 (cit. on p. 59).

[132] P. P. F. Barcelos, G. Guizzardi, A. S. Garcia, and M. E. Monteiro, "Ontological evaluation of the ITU-T Recommendation G.805," in *2011 18th International Conference on Telecommunications*, IEEE, May 2011, pp. 232–237, ISBN: 978-1-4577-0025-5. DOI: 10.1109/CTS.2011.5898926. [Online]. Available: http://dx.doi.org/10.1109/CTS.2011.5898926 (cit. on p. 59).

[133] A. Pras, J. Schonwalder, M. Burgess, O. Festor, G. Perez, R. Stadler, and B. Stiller, "Key research challenges in network management," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 104–110, Oct. 2007, ISSN: 0163-6804. DOI: 10.1109/MCOM.2007.4342832. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2007.4342832 (cit. on p. 59).

[134] M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. de Laat, "A Semantic-Web Approach for Modeling Computing Infrastructures," *Computers and Electrical Engineering*, vol. 39, no. 8, pp. 2553–2565, 2013. [Online]. Available: http://staff.science.uva.nl/~vdham/research/publications/1212-INDL-report.pdf (cit. on pp. 60, 179).

[135] P. S. Moraes, L. N. Sampaio, J. A. S. Monteiro, and M. Portnoi, "MonONTO: A Domain Ontology for Network Monitoring and Recommendation for Advanced Internet Applications Users," in *NOMS Workshops 2008 - IEEE Network Operations and Management Symposium Workshops*, IEEE, Apr. 2008, pp. 116–123, ISBN: 978-1-4244-2067-4. DOI: 10.1109/NOMSW.2007.21. [Online]. Available: http://dx.doi.org/10.1109/NOMSW.2007.21 (cit. on p. 60).

[136] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, Dec. 2012, ISSN: 15708268. DOI: 10.1016/j.websem.2012.05.003. [Online]. Available: http://dx.doi.org/10.1016/j.websem.2012.05.003 (cit. on pp. 60, 183).

[137] C. Shannon, D. Moore, K. Keys, M. Fomenkov, B. Huffaker, and K. Claffy, "The internet measurement data catalog," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, p. 97, Oct. 2005, ISSN: 01464833. DOI: 10.1145/1096536.1096552. [Online]. Available: http://dx.doi.org/10.1145/1096536.1096552 (cit. on p. 60).

[138] K. Alhamazani, R. Ranjan, K. Mitra, F. Rabhi, P. P. Jayaraman, S. U. Khan, A. Guabtni, and V. Bhatnagar, "An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art," *Computing*, vol. 97, no. 4, pp. 357–377, Apr. 2014, ISSN: 0010-485X. DOI: 10.1007/s00607-014-0398-5. [Online]. Available: http://dx.doi.org/10.1007/s00607-014-0398-5 (cit. on pp. 62, 63, 68–70).

[139] G. Carella, T. Magedanz, K. Campowsky, and F. Schreiner, "Elasticity as a service for federated cloud testbeds," in *2013 IEEE International Conference on Communications Workshops (ICC)*, IEEE, Jun. 2013, pp. 256–260, ISBN: 978-1-4673-5753-1. DOI: 10.1109/ICCW.2013.6649239. [Online]. Available: http://dx.doi.org/10.1109/ICCW.2013.6649239 (cit. on pp. 63, 138).

[140] J. Mwangama, N. Ventura, A. Willner, Y. Al-Hazmi, G. Carella, and T. Magedanz, "Towards Mobile Federated Network Operators," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, Apr. 2015, pp. 1–6, ISBN: 978-1-4799-7899-1. DOI: 10.1109/NETSOFT.2015.7116187. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7116187 (cit. on p. 63).

[141] Alcatel-Lucent, "White paper: Network Functions Virtualization – Challenges and Solutions.," 2013, [Online]. Available: http://www.tmcnet.com/tmc/

whitepapers/documents/whitepapers/2013/9377-network-functions-virtualization-challenges-solutions.pdf (cit. on p. 63).

[142] S. Fdida, T. Korakis, H. Niavis, S. Salsano, and G. Siracusano, "The EXPRESS SDN experiment in the OpenLab large scale shared experimental Facility," in *2014 First International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*, IEEE, Oct. 2014, pp. 1–7, ISBN: 978-1-4799-7595-2. DOI: 10.1109/MoNeTeC.2014.6995584. [Online]. Available: http://dx.doi.org/10.1109/MoNeTeC.2014.6995584 (cit. on p. 63).

[143] Y. Al-Hazmi and T. Magedanz, "Monitoring and Measurement Architecture for Federated Future Internet Experimentation Facilities," in *2014 European Conference on Networks and Communications (EuCNC)*, IEEE, IEEE, Jun. 2014, pp. 1–6, ISBN: 978-1-4799-5280-9. DOI: 10.1109/EuCNC.2014.6882663. [Online]. Available: http://dx.doi.org/10.1109/EuCNC.2014.6882663 (cit. on pp. 63, 64, 69, 70, 88, 91, 92, 128, 129, 133, 145).

[144] J. Augé, T. Parmentelat, N. Turro, S. Avakian, L. Baron, M. A. Larabi, M. Y. Rahman, T. Friedman, and S. Fdida, "Tools to foster a global federation of testbeds," *Computer Networks*, vol. 63, pp. 205–220, Apr. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.12.038. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.12.038 (cit. on p. 64).

[145] A. Gavras, H. Hrasnica, S. Wahle, D. Lozano, D. Mischler, and S. Denazis, "Control of resources in Pan-European testbed federation," in *Towards the Future Internet: A European Research Perspective*, IOS Press, 2009, pp. 67–78. DOI: 10.3233/978-1-60750-007-0-67. [Online]. Available: http://dx.doi.org/10.3233/978-1-60750-007-0-67 (cit. on pp. 64, 65, 70).

[146] O. Mehani, G. Jourjon, J. White, T. Rakotoarivelo, R. Boreli, and T. Ernst, "Characterisation of the Effect of a Measurement Library on the Performance of Instrumented Tools," Tech. rep. 4879. NICTA, Tech. Rep., 2011. [Online]. Available: http://nicta.com.au/pub-download/full/4879/ (cit. on pp. 67, 71).

[147] A. Gavras, A. Bak, G. Biczók, P. Gajowniczek, A. Gulyás, H. Hrasnica, P. Martinez-Julia, F. Németh, C. Papagianni, S. Papavassiliou, M. Pilarski, and A. Skarmeta, "Heterogeneous Testbeds, Tools and Experiments –Measurement Requirements Perspective," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7586 LNCS, 2013, pp. 139–158. DOI: 10.1007/978-3-642-41296-7_9. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41296-7_9 (cit. on pp. 67, 71).

[148] L. Fàbrega, V. Pere, D. Careglio, and D. Papadimitriou, *Measurement Methodology and Tools*, L. Fàbrega, P. Vilà, D. Careglio, and D. Papadimitriou, Eds., ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7586, ISBN: 978-3-642-41295-0. DOI: 10.1007/978-3-642-41296-7. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41296-7 (cit. on pp. 67, 69, 71).

[149] Y. Al-Hazmi, J. Gonzalez, P. Rodriguez-Archilla, F. Alvarez, T. Orphanoudakis, P. Karkazis, and T. Magedanz, "Unified Representation of Monitoring Information Across Federated Cloud Infrastructures," in *2014 26th International Teletraffic Congress (ITC)*, IEEE, IEEE, Sep. 2014, pp. 1–6, ISBN: 978-0-9883045-0-5. DOI: 10.1109/ITC.2014.6932978. [Online]. Available: http://dx.doi.org/10.1109/ITC.2014.6932978 (cit. on pp. 69–71, 143, 144, 173).

[150] S. Zanikolas and R. Sakellariou, "A taxonomy of grid monitoring systems," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 163–188, Jan. 2005, ISSN: 0167739X. DOI: 10.1016/j.future.2004.07.002. [Online]. Available: http://dx.doi.org/10.1016/j.future.2004.07.002 (cit. on pp. 69–71, 73).

[151] B. Parak and Z. Ustr, "Challenges in Achieving IaaS Cloud Interoperability across Multiple Cloud Management Frameworks," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, IEEE, Dec. 2014, pp. 404–411, ISBN: 978-1-4799-7881-6. DOI: 10.1109/UCC.2014.51. [Online]. Available: http://dx.doi.org/10.1109/UCC.2014.51 (cit. on p. 71).

[152] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013, ISSN: 15308669. DOI: 10.1002/wcm.1203. [Online]. Available: http://dx.doi.org/10.1002/wcm.1203 (cit. on p. 72).

[153] J. A. L. del Castillo, K. Mallichan, and Y. Al-Hazmi, "OpenStack Federation in Experimentation Multi-cloud Testbeds," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, IEEE, Dec. 2013, pp. 51–56, ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.103. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.103 (cit. on p. 83).

[154] P. Spyns, R. Meersman, and M. Jarrar, "Data modelling versus ontology engineering," *ACM SIGMOD Record*, vol. 31, no. 4, p. 12, Dec. 2002, ISSN: 01635808. DOI: 10.1145/637411.637413. [Online]. Available: http://dx.doi.org/10.1145/637411.637413 (cit. on p. 100).

[155] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, 1998, ISSN: 0169023X. DOI: 10.1016/S0169-023X(97)00056-6. [Online]. Available: http://dx.doi.org/10.1016/S0169-023X(97)00056-6 (cit. on pp. 100, XXXVII).

[156] J. E. López de Vergara, A. Guerrero, V. A. Villagrá, and J. Berrocal, "Ontology-Based Network Management: Study Cases and Lessons Learned," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 234–254, Sep. 2009, ISSN: 1064-7570. DOI: 10.1007/s10922-009-9129-1. [Online]. Available: http://dx.doi.org/10.1007/s10922-009-9129-1 (cit. on p. 101).

[157] M. A. Musen, "Dimensions of knowledge sharing and reuse," *Computers and Biomedical Research*, vol. 25, no. 5, pp. 435–467, Oct. 1992, ISSN: 00104809. DOI: 10.1016/0010-4809(92)90003-S. [Online]. Available: http://dx.doi.org/10.1016/0010-4809(92)90003-S (cit. on p. 101).

[158] N. Casellas, "Methodologies, Tools and Languages for Ontology Design," in *Legal Ontology Engineering*, ser. Law, Governance and Technology Series, vol. 3, Dordrecht: Springer Netherlands, 2011, ch. 3, pp. 57–107. DOI: 10.1007/978-94-007-1497-7_3. [Online]. Available: http://dx.doi.org/10.1007/978-94-007-1497-7_3 (cit. on p. 101).

[159] Y. Al-Hazmi and T. Magedanz, "MOFI: Monitoring Ontology for Federated Infrastructures," in *The 3rd IEEE International Workshop on Measurements & Networking (M&N) 2015*, Coimbra, Portugal: IEEE, 2015, pp. 140–145, ISBN: 978-1-4799-1860-7 (cit. on pp. 100, 104–108, 123, 153, 154, 180).

[160] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004, ISSN: 02767783. DOI: 10.2307/25148625. [Online]. Available: http://dl.acm.org/citation.cfm?id=2017217 (cit. on pp. 135, 136, 147, 168).

[161] K. Kavoussanakis, A. Hume, J. Martrat, C. Ragusa, M. Gienger, K. Campowsky, G. V. Seghbroeck, C. Vazquez, C. Velayos, F. Gittler, P. Inglesant, G. Carella, V. Engen, M. Giertych, G. Landi, and D. Margery, "BonFIRE: The Clouds and Services Testbed," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, IEEE, Dec. 2013, pp. 321–326, ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.156. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.156 (cit. on pp. 136, 137).

[162] R. Moreno-Vozmediano, Montero, and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, Dec. 2012, ISSN: 0018-9162. DOI: 10.1109/MC.2012.76. [Online]. Available: http://dx.doi.org/10.1109/MC.2012.76 (cit. on p. 136).

[163] A. Coles, E. Deliot, A. Edwards, A. Fischer, P. Goldsack, J. Guijarro, R. Hawkes, J. Kirschnick, S. Loughran, P. Murray, and L. Wilcock, "Cells: A Self-Hosting Virtual Infrastructure Service," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, IEEE, Nov. 2012, pp. 57–64, ISBN: 978-1-4673-4432-6. DOI: 10.1109/UCC.2012.17. [Online]. Available: http://dx.doi.org/10.1109/UCC.2012.17 (cit. on p. 136).

[164] P. Szegedi, J. Riera, J. Garcia-Espin, M. Hidell, P. Sjodin, P. Soderman, M. Ruffini, D. O'Mahony, A. Bianco, L. Giraudo, M. Ponce de Leon, G. Power, C. Cervello-Pastor, V. Lopez, and S. Naegele-Jackson, "Enabling future internet research: the FEDERICA case," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 54–61, Jul. 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5936155. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2011.5936155 (cit. on p. 137).

[165] A. Micsik, P. Pallinger, and D. Siklosi, "Scaling a Plagiarism Search Service on the BonFIRE Testbed," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, IEEE, Dec. 2013, pp. 57–62, ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.104. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.104 (cit. on p. 138).

[166] U. Wajid, C. A. Marín, and N. Mehandjiev, "Optimizing Service Ecosystems in the Cloud," in *Future Internet Assembly 2013: Validated Results and New Horizons*, 2013, pp. 115–126. DOI: 10.1007/978-3-642-38082-2_10. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38082-2_10 (cit. on p. 138).

[167] Y. Al-Hazmi and ..., "OpenLab: Interoperability of tools and data repositories," European FP7 Project OpenLabs, Tech. Rep., 2013, pp. 1–57. [Online]. Available: http://www.ict-openlab.eu/fileadmin/documents/public_deliverables/OpenLab_Deliverable_D2_2.pdf (cit. on p. 141).

[168] Y. Al-Hazmi, J. Müller, H. Coskun, and T. Magedanz, "Network Path and Quality Validation in the Evolved Packet Core," in *12th Würzburg Workshop on IP: ITG Workshop "Visions of Future Generation Networks" (EuroView2012), Würzburg, Germany*, 2012, pp. 1–2 (cit. on pp. 143, 174).

[169] J. Mueller, Y. Al-Hazmi, M. F. Sadikin, D. Vingarzan, and T. Magedanz, "Secure and Efficient Validation of Data Traffic Flows in Fixed and Mobile Networks," in *PM2HW2N'12 - Proceedings of the 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, 2012, pp. 159–165, ISBN: 978-1-4503-1626-2. DOI: 10.1145/2387191.2387213. [Online]. Available: http://dx.doi.org/10.1145/2387191.2387213 (cit. on pp. 143, 174).

[170] Y. Al-Hazmi, A. Willner, O. O. Ozpehlivan, D. Nehls, S. Covaci, and T. Magedanz, "An Automated Health Monitoring Solution for Future Internet Infrastructure Marketplaces," in *2014 26th International Teletraffic Congress (ITC)*, IEEE, Sep. 2014, pp. 1–6, ISBN: 978-0-9883045-0-5. DOI: 10.1109/ITC.2014.6932979. [Online]. Available: http://dx.doi.org/10.1109/ITC.2014.6932979 (cit. on p. 146).

[171] C. Bizer and A. Schultz, "The Berlin SPARQL Benchmark," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 2, pp. 1–24, 2009. [Online]. Available: http://libra.msra.cn/Publication/5502353/the-berlin-sparql-benchmark (cit. on p. 156).

[172] M. Voigt, A. Mitschick, and J. Schulz, "Yet Another Triple Store Benchmark? Practical Experiences with Real-World Data," in *2nd International Workshop on Semantic Digital Archives (SDA 2012)*, 2012, pp. 85–94 (cit. on p. 157).

[173] J. Mueller, D. Palma, G. Landi, J. Soares, B. Parreira, T. Metsch, P. Gray, A. Georgiev, Y. Al-Hazmi, T. Magedanz, and P. Simoes, "Monitoring as a Service for Cloud Environments," in *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, IEEE, Jul. 2014, pp. 174–179, ISBN: 978-1-4799-5051-5. DOI: 10.1109/CCE.2014.6916699. [Online]. Available: http://dx.doi.org/10.1109/CCE.2014.6916699 (cit. on p. 174).

[174] Storage Networking Industry Association (SNIA), "Cloud Data Management Interface (CDMI) V1.1.1," Storage Networking Industry Association (SNIA), Tech. Rep., 2015. [Online]. Available: http://www.snia.org/cdmi (cit. on pp. 27, 177).

[175] DMTF, *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol*, Sep. 2014 (cit. on pp. 27, 177).

[176] G. Iannaccone and C. Diot, "The CoMo Project: Towards A Community-Oriented Measurement Infrastructure," in *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, Boston: Kluwer Academic Publishers, 2006, pp. 97–111. DOI: 10.1007/0-387-30394-4_8. [Online]. Available: http://dx.doi.org/10.1007/0-387-30394-4_8 (cit. on pp. 48, 177).

[177] T. Faber and J. Wroclawski, "A federated experiment environment for emulab-based testbeds," in *2009 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities and Workshops, TridentCom 2009*, 2009. DOI: 10.1109/TRIDENTCOM.2009.4976238. [Online]. Available: http://dx.doi.org/10.1109/TRIDENTCOM.2009.4976238 (cit. on pp. 29, 178).

[178] Y. Shavitt and E. Shir, "DIMES: Let the Internet Measure Itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, p. 71, Oct. 2005, ISSN: 01464833. DOI: 10.1145/1096536.1096546. [Online]. Available: http://dx.doi.org/10.1145/1096536.1096546 (cit. on pp. 48, 178).

[179] D. Morato, E. Magana, M. Izal, J. Aracil, F. Naranjo, F. Astiz, U. Alonso, I. Csabai, P. Haga, G. Simon, J. Steger, and G. Vattay, "The European Traffic Observatory Measurement Infraestructure (ETOMIC): A Testbed for Universal Active and Passive Measurements," in *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, IEEE, 2005, pp. 283–289, ISBN: 0-7695-2219-X. DOI: 10.1109/TRIDNT.2005.34. [Online]. Available: http://dx.doi.org/10.1109/TRIDNT.2005.34 (cit. on pp. 48, 178).

[180] I. Csabai, A. Fekete, P. Hága, B. Hullár, G. Kurucz, S. Laki, P. Mátray, J. Stéger, G. Vattay, F. Espina, S. Garcia-Jimenez, M. Izal, E. Magaña, D. Morató, J. Aracil, F. Gómez, I. Gonzalez, S. López-Buedo, V. Moreno, and J. Ramos, "ETOMIC Advanced Network Monitoring System for Future Internet Experimentation," in *6th International Conference on Testbeds and Research Infrastructures. Development of Networks and Communities (Tridentcom2010)*, Springer Berlin Heidelberg, 2011, pp. 243–254. DOI: 10.1007/978-3-642-17851-1_20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17851-1_20 (cit. on pp. 48, 178).

[181] D. Havlik, S. Schade, W. van Wijk, T. Usländer, and J. Hierro, "Leveraging the Future Internet for the Environmental Usage Area," in *Innovations in Sharing Environmental Observations and Information*, Shaker Verlag Aachen, 2011, ISBN: 978-3-8440-0451-9. [Online]. Available: http://publications.jrc.ec.europa.eu/repository/handle/JRC66152 (cit. on pp. 2, 178).

[182] J. Crowcroft, P. Demeester, J. Magen, P. Tran-gia, and J. Wilander, "Towards a collaboration and high-level federation structure for the FIRE Facility," *Working Group on modular federation of FIRE Facilities*, 2009. [Online]. Available: http://initiative.future-internet.eu/uploads/media/Wise-men_final.pdf (cit. on pp. 35, 178).

[183] A. Willner, D. Nehls, and T. Magedanz, "FITeagle: Semantic Testbed Management Framework," in *10th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2015)0th Int. Conf. on Testbeds and Research Infrastr. for the Dev. of Netw. & Comm*, 2015, pp. 1–4. DOI: 10.4108/icst.tridentcom.2015.259748. [Online]. Available: http://eudl.eu/doi/10.4108/icst.tridentcom.2015.259748 (cit. on pp. 42, 178).

[184] A. Glikson, "FI-WARE: Core Platform for Future Internet Applications," in *4th Annual International Conference on Systems and Storage*, Haifa, 2011 (cit. on pp. 3, 178).

[185] C. Thuemmler, J. Mueller, S. Covaci, T. Magedanz, S. de Panfilis, T. Jell, and A. Gavras, "Applying the Software-to-Data Paradigm in Next Generation E-Health Hybrid Clouds," in *2013 10th International Conference on Information Technology: New Generations*, IEEE, Apr. 2013, pp. 459–463, ISBN: 978-0-7695-4967-5. DOI: 10.1109/ITNG.2013.77. [Online]. Available: http://dx.doi.org/10.1109/ITNG.2013.77 (cit. on pp. 33, 178).

[186] T. Rakotoarivelo, G. Jourjon, and M. Ott, "Designing and orchestrating reproducible experiments on federated networking testbeds," *Computer Networks*, vol. 63, pp. 173–187, Apr. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.12.033. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.12.033 (cit. on pp. 42, 179).

[187] D. Schwerdel, B. Reuther, T. Zinner, P. Müller, and P. Tran-Gia, "Future Internet research and experimentation: The G-Lab approach," *Computer Networks*, vol. 61, pp. 102–117, Mar. 2014, ISSN: 13891286. DOI: 10.1016/j.bjp.2013.12.023. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.12.023 (cit. on pp. 2, 179).

[188] J. Duerig, R. Ricci, L. Stoller, and M. Strum, "Getting started with geni: a user tutorial," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 1, pp. 72–77, 2012. DOI: 10.1145/2096149.2096161. [Online]. Available: http://dx.doi.org/10.1145/2096149.2096161 (cit. on pp. 2, 179).

[189] A. M. Alberti and D. Singh, "Internet of Things: Perspectives, Challenges and Opportunities," in *Int. Workshop on Telecommunications (IWT)*, 2013 (cit. on pp. 2, 179).

[190] G. Karagiannis, A. Jamakovic, A. Edmonds, C. Parada, T. Metsch, D. Pichon, M. Corici, S. Ruffino, A. Gomes, P. S. Crosta, and T. M. Bohnert, "Mobile Cloud Networking: Virtualisation of cellular networks," in *2014 21st International Conference on Telecommunications (ICT)*, IEEE, May 2014, pp. 410–415, ISBN: 978-1-4799-5141-3. DOI: 10.1109/ICT.2014.6845149. [Online]. Available: http://dx.doi.org/10.1109/ICT.2014.6845149 (cit. on pp. 174, 180).

[191] C. Brandauer and T. Fichtel, "MINER - A measurement infrastructure for network research," in *2009 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops*, IEEE, 2009, pp. 1–9, ISBN: 978-1-4244-2846-5. DOI: 10.1109/TRIDENTCOM.2009.4976235. [Online]. Available: http://dx.doi.org/10.1109/TRIDENTCOM.2009.4976235 (cit. on pp. 53, 180).

[192] A. Quereilhac, M. Lacage, C. Freire, T. Turletti, and W. Dabbous, "NEPI: An integration framework for Network Experimentation," *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–5, 2011. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.227.6433 (cit. on pp. 42, 180).

[193] ETSI-NFV, "European Telecommunications Standards Institute (ETSI) Industry Specification Group Network Functions Virtualisation (NFV), white paper," 2013, [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/689-network-functions-virtualizaation (cit. on pp. 2, 180).

[194] M. Bauer, E. Kovacs, A. Schulke, N. Ito, C. Criminisi, L.-W. Goix, and M. Valla, "The Context API in the OMA Next Generation Service Interface," in *2010 14th International Conference on Intelligence in Next Generation Networks*, IEEE, IEEE, Oct. 2010, pp. 1–5, ISBN: 978-1-4244-7443-1. DOI: 10.1109/ICIN.2010.5640931. [Online]. Available: http://dx.doi.org/10.1109/ICIN.2010.5640931 (cit. on pp. 144, 181).

[195] J. van der Ham, F. Dijkstra, R. Lapacz, and J. Zurawski, *GFD.206: Network markup language base schema version 1*, 2013. [Online]. Available: http://www.ogf.org/documents/GFD.206.pdf (cit. on pp. 59, 181).

[196] P. Mátray, I. Csabai, P. Hága, J. Stéger, L. Dobos, and G. Vattay, "Building a prototype for network measurement virtual observatory," *Proceedings of the 3rd annual ACM workshop on Mining network data - MineNet '07*, p. 23, 2007. DOI: 10.1145/1269880.1269887. [Online]. Available: http://dx.doi.org/10.1145/1269880.1269887 (cit. on pp. 48, 181).

[197] T. Metsch and A. Edmonds, "Open Cloud Computing Interface–Infrastructure," in *Standards Track, no. GFD-R in The Open Grid Forum Document Series, Open Cloud Computing Interface (OCCI) Working Group, Muncie (IN)*, Open Grid Forum (OGF), 2010. [Online]. Available: http://occi-wg.org/about/specification/ (cit. on pp. 26, 181).

[198] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, p. 54, Jan. 2010, ISSN: 01635980. DOI: 10.1145/1713254.1713267. [Online]. Available: http://dx.doi.org/10.1145/1713254.1713267 (cit. on pp. 42, 181).

[199] M. Singh, M. Ott, I. Seskar, and P. Kamat, "ORBIT Measurements framework and library (OML): motivations, implementation and features," in *IEEE Tridentcom: First Int. Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2005, pp. 146–152. DOI:

10.1109/TRIDNT.2005.25. [Online]. Available: http://dx.doi.org/10.1109/TRIDNT.2005.25 (cit. on pp. 42, 181).

[200] S. Fdida, T. Friedman, and T. Parmentelat, "OneLab: An Open Federated Facility for Experimentally Driven Future Internet Research," in *New Network Architectures*, vol. 297, 2010, pp. 141–152. DOI: 10.1007/978-3-642-13247-6_7. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13247-6_7 (cit. on pp. 30, 181).

[201] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, *A one-way active measurement protocol (OWAMP)*, 2006. [Online]. Available: https://tools.ietf.org/html/rfc4656 (cit. on pp. 49, 181).

[202] M. Corici, F. Gouveia, T. Magedanz, and D. Vingarzan, "OpenEPC: A Technical Infrastructure for Early Prototyping of NGMN Testbeds," in, 2011, pp. 166–175. DOI: 10.1007/978-3-642-17851-1_13. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17851-1_13 (cit. on pp. 143, 181).

[203] T. Friedman and A. Gavras, "FIRE OpenLab IP Testbed and Tool Demo," in *ServiceWave 2011, LNCS*, vol. 6994, Springer-Verlag Berlin Heidelberg, 2011, pp. 323–324. DOI: 10.1007/978-3-642-24755-2_36. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24755-2_36 (cit. on pp. 30, 181).

[204] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3, IEEE, 2005, pp. 1664–1669, ISBN: 0-7803-8966-2. DOI: 10.1109/WCNC.2005.1424763. [Online]. Available: http://dx.doi.org/10.1109/WCNC.2005.1424763 (cit. on pp. 29, 182).

[205] J. Chase, L. Grit, and D. Irwin, "Beyond virtual data centers: Toward an open resource control architecture," in *Selected Papers from the Int. Conf. on the Virtual Computing Initiative (ACM Digital Library)*, 2007. [Online]. Available: http://www.cs.duke.edu/nicl/pub/papers/vci07.pdf (cit. on pp. 29, 182).

[206] Dmtf, "Open Virtualization Format Specification," *DMTF Virtualization Management VMAN Initiative*, pp. 1–42, 2010. [Online]. Available: http://www.dmtf.org/standards/ovf (cit. on pp. 27, 182).

[207] S. Wahle, T. Magedanz, S. Fox, and E. Power, "Heterogeous resource description and management in generic resource federation frameworks," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, IEEE, May 2011, pp. 1196–1199, ISBN:

978-1-4244-9219-0. DOI: 10.1109/INM.2011.5990582. [Online]. Available: http://dx.doi.org/10.1109/INM.2011.5990582 (cit. on pp. 35, 182).

[208] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences Building PlanetLab," *Symposium A Quarterly Journal In Modern Foreign Literatures*, vol. 19, no. 1, pp. 351–366, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1298455.1298489 (cit. on pp. 2, 182).

[209] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, p. 3, Jul. 2003, ISSN: 01464833. DOI: 10.1145/956993.956995. [Online]. Available: http://portal.acm.org/citation.cfm?doid=956993.956995 (cit. on pp. 2, 182).

[210] The GENI Project Office, "ProtoGENI Control Framework Overview," 2009, [Online]. Available: http://groups.geni.net/geni/wiki/GeniControl (cit. on pp. 29, 182).

[211] T. Santos, C. Henke, C. Schmoll, and T. Zseby, "Multi-hop packet tracking for experimental facilities," in *ACM SIGCOMM 2010 conference*, 2010, pp. 447–448. DOI: 10.1145/1851275.1851256. [Online]. Available: http://dx.doi.org/10.1145/1851275.1851256 (cit. on pp. 48, 182).

[212] D. Brickley and R. V. Guha, "Resource Description Framework (RDF) Schema 1.1. W3C Recommendation," *World Wide Web Consortium*, 2014. [Online]. Available: http://www.w3.org/TR/rdf-schema/ (cit. on pp. 58, 182).

[213] G. Klyne, J. J. Carroll, and B. McBride, "Resource description framework (RDF): Concepts and abstract syntax," W3C, W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/rdf-concepts/ (cit. on pp. 58, 182).

[214] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *Internet Computing, IEEE*, vol. 11, no. 6, pp. 60–67, 2007. [Online]. Available: http://www.w3.org/TR/sawsdl/ (cit. on pp. 59, 182).

[215] Open Networking Foundation (ONF), "SDN Architecture," Tech. Rep., 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN-ARCH-Overview-1.1-11112014.02.pdf (cit. on pp. 2, 182).

[216] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001, ISSN: 0036-8733. DOI: 10.1038/scientificamerican0501-34. [Online]. Available: http://dx.doi.org/10.1038/scientificamerican0501-34 (cit. on pp. 56, 183).

[217] L. Peterson, S. Sevinc, J. Lepreau, and R. Ricci, "Slice-based Federation architecture," GENI, Tech. Rep., 2009. [Online]. Available: http://groups.geni.net/geni/wiki/SliceFedArch (cit. on pp. 42, 183).

[218] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing SNMP management frameworks," *Rfc 2571*, p. 62, 1999. [Online]. Available: https://www.ietf.org/rfc/rfc2571.txt (cit. on pp. 48, 183).

[219] B. Hullár, S. Laki, J. Stéger, I. Csabai, and G. Vattay, "SONoMA: A service oriented network measurement architecture," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 90 LNICST, 2012, pp. 27–42, ISBN: 9783642292729. DOI: 10.1007/978-3-642-29273-6_3. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29273-6_3 (cit. on pp. 48, 183).

[220] E. Prud'Hommeaux, A. Seaborne, and E. Al., "SPARQL Query Language for RDF," *W3C recommendation*, 2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/ (cit. on pp. 124, 183).

[221] S. Wahle, C. Tranoris, S. Denazis, A. Gavras, K. Koutsopoulos, T. Magedanz, and S. Tompros, "Emerging testing trends and the Panlab enabling infrastructure," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 167–175, Mar. 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5723816. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2011.5723816 (cit. on pp. 42, 129, 183).

[222] D. Palma and T. Spatzier, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1*, Nov. 2013. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html (cit. on pp. 27, 183).

A

Author's Peer-Reviewed Publications

[1] M. Morsey, Y. Al-Hazmi, M. Giatili, C. Papagianni, P. Grosso, I. Baldin, and A. Willner, "Open-Multinet: Supporting Resource Management in Federated Infrastructures with Semantic Web Ontologies", in *25rd International World Wide Web Conference (WWW 2016)*, ACM, 2015, pp. 1-12. (under submission).

[2] Y. Al-Hazmi and T. Magedanz, "MOFI: Monitoring Ontology for Federated Infrastructures", in *The 3rd IEEE International Workshop on Measurements & Networking (M&N) 2015*, Coimbra, Portugal: IEEE, 2015, pp. 140-145, ISBN: 978-1-4799-1860-7.

[3] Y. Al-Hazmi and T. Magedanz, "Towards Semantic Monitoring Data Collection and Representation in Federated Infrastructures", in *3rd International Conference on Future Internet of Things and Cloud (FiCloud2015)*, IEEE, 2015, pp. 17-24. DOI: 10.1109/FiCloud.2015.40 [Online]. Available: http://dx.doi.org/10.1109/FiCloud.2015.40.

[4] Y. Al-Hazmi, A. Willner, B. Pickering, A. Alloush, T. Magedanz, and S. Cretti, "Creating a Sustainable Federation of Cloud-Based Infrastructures for the Future Internet –The FIWARE Approach", in *10th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2015)*, Vancouver, Canada: ACM, 2015, pp. [Online]. Available: http://dx.doi.org/10.4108/icst.tridentcom.2015.259747

[5] A. Willner, C. Papagianni, M. Giatili, P. Grosso, M. Morsey, Y. Al-Hazmi, and I. Baldin, "Open-Multinet Upper Ontology –Towards the Semantic-based Management of Federated Infrastructures", in *10th International Conference TRIDENTCOM 2015*, Vancouver, Canada: ACM, 2015, pp. 1-10. [Online]. Available: http://dx.doi.org/10.4108/icst.tridentcom.2015.259750

[6] J. Mwangama, N. Ventura, A. Willner, Y. Al-Hazmi, G. Carella, and T. Magedanz, "Towards Mobile Federated Network Operators", in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, Apr. 2015, pp. 1-6, ISBN: 978-1-4799-7899-1. [Online]. Available: http://dx.doi.org/10.1109/NETSOFT.2015.7116187

[7] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, and Y. Al-Hazmi, "FIWARE Lab: Managing Resources and Services in a Cloud Federation Supporting Future Internet Applications", in *7th International Conference on Utility and Cloud Computing*, London, UK: IEEE/ACM, 2014. [Online]. Available: http://dx.doi.org/10.1109/UCC.2014.129

[8] Y. Al-Hazmi, J. Gonzalez, P. Rodriguez-Archilla, F. Alvarez, T. Orphanoudakis, P. Karkazis, and T. Magedanz, "Unified Representation of Monitoring Information Across Federated Cloud Infrastructures", in *2014 26th International Teletraffic Congress (ITC)*, IEEE, IEEE, Sep. 2014, pp. 1-6, ISBN: 978-0-9883045-0-5. [Online]. Available: http://dx.doi.org/10.1109/ITC.2014.6932978

[9] Y. Al-Hazmi, A. Willner, O. O. Ozpehlivan, D. Nehls, S. Covaci, and T. Magedanz, "An Automated Health Monitoring Solution for Future Internet Infrastructure Marketplaces", in *2014 26th International Teletraffic Congress (ITC)*, IEEE, Sep. 2014, pp. 1-6, ISBN: 978-0-9883045-0-5. [Online]. Available: http://dx.doi.org/10.1109/ITC.2014.6932979

[10] J. Mueller, D. Palma, G. Landi, J. Soares, B. Parreira, T. Metsch, P. Gray, A. Georgiev, Y. Al-Hazmi, T. Magedanz, and P. Simoes, "Monitoring as a Service for Cloud Environments", in *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, IEEE, Jul. 2014, pp. 174-179, ISBN: 978-1-4799-5051-5. [Online]. Available: http://dx.doi.org/10.1109/CCE.2014.6916699

[11] Y. Al-Hazmi and T. Magedanz, "Monitoring and Measurement Architecture for Federated Future Internet Experimentation Facilities", in *2014 European Conference on Networks and Communications (EuCNC)*, IEEE, Jun. 2014, pp. 1-6, ISBN: 978-1-4799-5280-9. [Online]. Available: http://dx.doi.org/10.1109/EuCNC.2014.6882663

[12] T. Wauters, B. Vermeulen, W. Vandenberghe, P. Demeester, S. Taylor, L. Baron, M. Smirnov, Y. Al-Hazmi, A. Willner, M. Sawyer, D. Margery, T. Rakotoarivelo, F. L. Vilela, D. Stavropoulos, C. Papagianni, F. Francois, C. Bermudo, A. Gavras, D. Davies, J. Lanza, and S.-Y. Park, "Federation of Internet Experimentation Facilities: Architecture and Implementation ", in

European Conference on Networks and Communications, IEEE, Jun. 2014, pp. 1-5. [Online]. Available: http://hdl.handle.net/1854/LU-5732987

[13] J. A. L. del Castillo, K. Mallichan, and Y. Al-Hazmi, "OpenStack Federation in Experimentation Multi-cloud Testbeds", in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, IEEE, Dec. 2013, pp. 51-56, ISBN: 978-0-7695-5095-4. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.103

[14] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, A. Quereilhac, Y. Al-Hazmi, F. Lobillo, C. Velayos, A. Vicooton, and G. Androulidakis, "Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities", in *Future Network and Mobile Summit (FNMS)*, Lisboa, Portugal: IEEE, 2013, pp. 1-11, ISBN: 978-1-905824-37-3. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6633558

[15] D. García-Pérez, J. Á. L. del Castillo, Y. Al-Hazmi, J. Martrat, K. Avoussanakis, A. C. Hume, C. Velayos López, G. Landi, T. Wauters, M. Gienger, and D. Margery, "Cloud and Network Facilities Federation in BonFIRE", in *Federative and interoperable cloud infrastructures*, Aug 2013, Aachen, Germany. Euro-Par 2013: Parallel Processing Workshops, 8374, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 126-135. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54420-0_13

[16] Y. Al-Hazmi and T. Magedanz, "A Flexible Monitoring System for Federated Future Internet Testbeds", in *2012 Third International Conference on The Network of the Future (NOF)*, IEEE, Nov. 2012, pp. 1-6, ISBN: 978-1-4673-5265-9. [Online]. Available: http://dx.doi.org/10.1109/NOF.2012.6463985

[17] Y. Al-Hazmi, K. Campowsky, and T. Magedanz, "A Monitoring System for Federated Clouds", in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, IEEE, Nov. 2012, pp. 68-74, ISBN: 978-1-4673-2798-5. [Online]. Available: http://dx.doi.org/10.1109/CloudNet.2012.6483657

[18] J. Mueller, Y. Al-Hazmi, M. F. Sadikin, D. Vingarzan, and T. Magedanz, "Secure and Efficient Validation of Data Traffic Flows in Fixed and Mobile Networks", in *PM2HW2N'12 - Proceedings of the 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, 2012, pp. 159-165, ISBN: 978-1-4503-1626-2. [Online]. Available: http://dx.doi.org/10.1145/2387191.2387213

[19] Y. Al-Hazmi, J. Mueller, H. Coskun, and T. Magedanz, "Network Path and Quality Validation in the Evolved Packet Core", in *12th Würzburg Work-*

shop on IP: ITG Workshop "Visions of Future Generation Networks" (EuroView2012), Würzburg, Germany, 2012, pp. 1-2. [Online]. Available: http://www.euroview2012.org/fileadmin/content/euroview2012/abstracts/03_07_abstract_alhazmi.pdf

[20] A. C. Hume, Y. Al-Hazmi, B. Belter, K. Campowsky, L. M. Carril, G. Carrozzo, V. Engen, D. García-Pérez, J. Jofre Ponsatí, R. Kubert, Y. Liang, C. Rohr, and G. Van Seghbroeck, "BonFIRE: A Multi-cloud test Facility for Internet of Services Experimentation", in *8th International ICST Conference, TridentCom 2012*, vol. 44 LNICST, 2012, pp. 81-96. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35576-9_11

[21] S. Sargento, R. Matos, K. A. Hummel, A. Hess, S. Toumpis, Y. Tselekounis, G. D. Stamoulis, Y. Al-Hazmi, M. Ali, and H. de Meer, "Multi-Access Communications in Wireless Mesh Networks by Virtualization", in Wireless Multi-Access Environments and Quality of Service Provisioning, IGI Global, 2011, pp. 97-138, ISBN: 9781466600171. [Online]. Available: http://dx.doi.org/10.4018/978-1-4666-0017-1.ch005.

[22] Y. Al-Hazmi, H. de Meer, K. A. Hummel, H. Meyer, M. Meo, and D. Remondo, "Energy-Efficient Wireless Mesh Infrastructures", *IEEE Network*, vol. 25, no. 2, pp. 32-38, Mar. 2011, issn: 0890-8044. [Online]. Available: http://dx.doi.org/10.1109/MNET.2011.5730526

[23] Y. Al-Hazmi and H. de Meer, "Virtualization of 802.11 Interfaces for Wireless Mesh Networks", in *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, IEEE, Jan. 2011, pp. 44-51, ISBN: 978-1-61284-189-2. [Online]. Available: http://dx.doi.org/10.1109/WONS.2011.5720199

[24] A. Galis, S. Clayman, A. Fischer, A. Paler, Y. Al-Hazmi, H. De Meer, A. Cheniour, O. Mornard, J. P. Gelas, L. Lefevre, J. R. Loyola, A. Astorga, J. Serrat, and S. Davy, "Future Internet Management Platforms for Network Virtualisation and Service Clouds", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6481 LNCS, 2010, pp. 235-237. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17694-4_39

[25] I. Coskun, Hakan Schieferdecker and Y. Al-Hazmi, "Virtual WLAN: Going beyond Virtual Access Points", *Electronic Communications of the EASST*, vol. 25, no. 2, pp. 32-38, 2009, issn: 1863-2122. [Online]. Available: http://journal.ub.tu-berlin.de/eceasst/article/view/226

B

Monitoring Ontologies

This section presents some of the concepts and relations modelled in the MOFI ontologies written in Turtle. Note that only parts of the ontologies (one fifth of their classes and properties) are presented, just to give an idea of these ontologies. Furthermore, the Uniform Resource Name (URN) used below in the ontology prefixes is the MOFI namespace, however, the OMN namespace has been used in the implementation. Finally, the MOFI Unit ontology is not included as the NOVI Unit ontology has been adopted as is.

Listing B.1 includes only some classes and properties of the MOFI upper ontology.

Listing B.1: Part of the MOFI Upper ontology

```
@prefix : <urn:mofi#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix omn: <http://open-multinet.info/ontology/omn#> .
@prefix omn-service: <http://open-multinet.info/ontology/omn-service#> .
@prefix omn-resource: <http://open-multinet.info/ontology/omn-resource#> .
@prefix omn-component: <http://open-multinet.info/ontology/omn-component#> .
@prefix omn-lifecycle: <http://open-multinet.info/ontology/omn-lifecycle#> .
@prefix omn-federation: <http://open-multinet.info/ontology/omn-federation#> .
@base <urn:mofi> .

<urn:mofi> rdf:type owl:Ontology ;

          rdfs:label "mofi"@en ;
          dc:title "MOFI␣Upper␣Ontology"^^xsd:string ;
          dc:description "This␣ontology␣includes␣the␣main␣monitoring␣concepts␣and␣
              relations."^^xsd:string ;
          vann:preferredNamespacePrefix "mofi" ;
          dc:date "2015-05-30" ;
```

```
24          dc:creator <https://www.linkedin.com/in/yahyaalhazmi> ;
25          dc:author <https://www.linkedin.com/in/yahyaalhazmi> .
26
27  ###################################################################
28  # Object Properties
29  ###################################################################
30
31  ### urn:mofi#isMeasurementMetricOf
32  :isMeasurementMetricOf rdf:type owl:ObjectProperty ;
33
34                      rdfs:comment "Is␣a␣measurement␣metric␣of␣any␣observed␣object,␣e.g.␣
                            resource,␣component␣or␣service"^^xsd:string ;
35
36                      owl:inverseOf :hasMeasurementMetric ;
37
38                      rdfs:domain :Metric ;
39
40                      rdfs:range [ rdf:type owl:Class ;
41                                   owl:unionOf ( omn:Component
42                                                 omn:Resource
43                                                 omn:Service
44                                               )
45                                 ] .
46
47  ### urn:mofi-metric#hasMeasurementMetric
48  :hasMeasurementMetric rdf:type owl:ObjectProperty ;
49
50                      owl:inverseOf :isMeasurementMetricOf ;
51
52                      rdfs:range :Metric ;
53
54                      rdfs:domain [ rdf:type owl:Class ;
55                                    owl:unionOf ( omn:Component
56                                                  omn:Resource
57                                                  omn:Service
58                                                )
59                                  ] .
60
61  ### urn:mofi#isMeasurementOf
62  :isMeasurementOf rdf:type owl:ObjectProperty ;
63
64                  rdfs:comment "Defines␣the␣relation␣between␣a␣measurement␣and␣a␣metric"^^
                        xsd:string ;
65
66                  rdfs:domain :Measurement ;
67
68                  rdfs:range :Metric .
69
70  ### urn:mofi#measuredBy
71  :measuredBy rdf:type owl:ObjectProperty ;
72
73              owl:inverseOf :measuresMetric ;
74
75              rdfs:domain :Metric ;
76
77              rdfs:range :Tool .
78
```

```
### urn:mofi#hasUnit
:hasUnit rdf:type owl:ObjectProperty ;

        rdfs:domain :Data ;

        rdfs:range :Unit .

### urn:mofi#sentFrom
:sentFrom rdf:type owl:ObjectProperty ;

         rdfs:domain [ rdf:type owl:Class ;
                      owl:unionOf ( :Data
                                    omn-lifecycle:Request
                                  )
                    ] ;

         rdfs:range [ rdf:type owl:Class ;
                      owl:unionOf ( :Tool
                                    omn-federation:Infrastructure
                                  )
                    ] .

#################################################################
# Data properties
#################################################################

### urn:mofi#isRequested
:isRequested rdf:type owl:DatatypeProperty ;

            rdfs:comment "Indicate if a service is requested (True) or not (False)."^^
                 xsd:string ;

            rdfs:domain :MonitoringService ;

            rdfs:range xsd:boolean .

#################################################################
# Classes
#################################################################

### urn:mofi#MonitoringService
:MonitoringService rdf:type owl:Class ;

                rdfs:subClassOf omn:Service ;

                rdfs:comment "Rrepresents any monitoring service offered."^^xsd:string .


### urn:mofi#InfrastructureHealthMonitoring
:InfrastructureHealthMonitoring rdf:type owl:Class ;

                            rdfs:subClassOf :MonitoringService ;

                            rdfs:comment "Rrepresents the monitoring service that gives
                                 high level monitoring information about the health and
                                 the status of an ICT infrastructure."^^xsd:string .
```

```
### urn:mofi#InfrastructureResourceMonitoring
:InfrastructureResourceMonitoring rdf:type owl:Class ;

                                rdfs:subClassOf :MonitoringService ;

                                rdfs:comment "Rrepresents␣the␣monitoring␣service␣that␣
                                    gives␣detailed␣monitoring␣information␣about␣the␣used
                                    ␣resources␣and␣services␣at␣an␣ICT␣infrastructure."^^
                                    xsd:string .

### urn:mofi#SLAMonitoring
:SLAMonitoring rdf:type owl:Class ;

             rdfs:subClassOf :MonitoringService ;

             rdfs:comment "Rrepresents␣the␣Service␣Level␣Agreement␣(SLA)␣monitoring␣
                 service␣that␣gives␣detailed␣monitoring␣information␣about␣pre-defined␣
                 metrics␣to␣observe␣the␣SLAs␣whether␣they␣are␣met␣or␣violated."^^xsd:
                 string .

### urn:mofi#Data
:Data rdf:type owl:Class ;

     rdfs:comment "Data␣represents␣measurement␣data␣as␣well␣as␣other␣monitoring␣related␣
         information."^^xsd:string .

### urn:mofi#Metric
:Metric rdf:type owl:Class ;

       rdfs:comment "Metric␣is␣anything␣that␣can␣be␣measured,␣such␣as␣CPU␣load␣of␣a␣
           machine,␣packet␣loss␣in␣a␣channel,␣etc."^^xsd:string .

### urn:mofi#Tool
:Tool rdf:type owl:Class ;

     rdfs:comment "Tool␣represents␣tools␣used␣for␣performing␣measurements␣and␣monitoring
         ␣such␣as␣measurement␣probes,␣data␣collectors,␣visualizer,␣etc."^^xsd:string ;

     rdfs:subClassOf omn:Resource .
                    [ rdf:type owl:Restriction ;
                      owl:onProperty omn:hasService ;
                      owl:onClass omn:Service ;
                      owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger
                    ] .

### urn:mofi#Unit
:Unit rdf:type owl:Class ;

     rdfs:comment "Unit␣represents␣the␣unit␣of␣the␣measurement␣and␣monitoring␣data␣such␣
         as␣Bytes,␣Bitspersecond,␣etc."^^xsd:string .

### urn:mofi#Lifetime
:Lifetime rdf:type owl:Class ;

         rdfs:comment "Represents␣lifetime␣of␣any␣process␣(e.g.␣monitoring␣service␣or␣
             measurement)."^^xsd:string .
```

Listing B.2 shows only a very limited number of classes and properties from the MOFI Metric ontology. This listing has been shortened by removing some prefixes of external imported ontologies and metadata information, in contrast to Listing B.1 (prefixes in lines 2 to 15 and the metadata in lines 20 to 29). However, this ontology imports all these prefixes in addition to those of the other MOFI ontologies.

Listing B.2: Part of the MOFI Metric ontology

```
1  @prefix : <urn:mofi-metric#> .
2  ...
3  @prefix mofi: <urn:mofi#> .
4  @prefix mofi-metric: <urn:mofi-data#> .
5  @prefix mofi-unit: <urn:mofi-unit#> .
6  @prefix mofi-tool: <urn:mofi-tool#> .
7  @prefix mofi-genericconcepts: <urn:mofi-genericconcepts#> .
8  @base <urn:mofi-metric> .
9
10 <urn:mofi-metric> rdf:type owl:Ontology ;
11
12                 rdfs:label "mofi-metric"@en .
13
14 ###################################################################
15 # Object Properties
16 ###################################################################
17
18 ### urn:mofi-metric#canBeCalculatedFrom
19 :canBeCalculatedFrom rdf:type owl:ObjectProperty ;
20
21                   rdfs:domain mofi:Metric ;
22
23                   rdfs:range mofi:Metric .
24
25 ###################################################################
26 # Data properties
27 ###################################################################
28
29 ### urn:mofi-metric#hasFrequency
30 :hasFrequency rdf:type owl:DatatypeProperty ;
31
32             rdfs:comment "Presents␣the␣updating␣rate␣of␣the␣measurements,␣e.g.␣if␣the␣
                   frequency␣is␣set␣to␣30␣second,␣i.e.␣a␣new␣measure␣is␣provided␣every␣30␣
                   second."^^xsd:string ;
33
34             rdfs:domain mofi:Metric ;
35
36             rdfs:range xsd:integer .
37
38 ###################################################################
39 # Classes
40 ###################################################################
41
42 ### urn:mofi-metric#Availability
43 :Availability rdf:type owl:Class ;
44
45             rdfs:subClassOf mofi:Metric ;
46
```

```
47              rdfs:comment "Represents␣resource␣availability␣status,␣1=available,␣0=not,␣
                    2=anything␣else␣(in␣Maintenance)."^^xsd:string .
48
49  ### urn:mofi-metric#BandwidthUtilization
50  :BandwidthUtilization rdf:type owl:Class ;
51
52                      rdfs:subClassOf mofi:Metric .
53
54  ### urn:mofi-metric#CPULoad
55  :CPULoad rdf:type owl:Class ;
56
57          rdfs:subClassOf :CPUUtilization .
58
59  ### urn:mofi-metric#CPUUtilization
60  :CPUUtilization rdf:type owl:Class ;
61
62                rdfs:subClassOf mofi:Metric .
63
64  ### urn:mofi-metric#Delay
65  :Delay rdf:type owl:Class ;
66
67        rdfs:subClassOf mofi:Metric .
68
69  ### urn:mofi-metric#InterferenceLevel
70  :InterferenceLevel rdf:type owl:Class ;
71
72                  rdfs:subClassOf :RadioSignalQuality .
73
74  ### urn:mofi-metric#OneWayDelay
75  :OneWayDelay rdf:type owl:Class ;
76
77              rdfs:subClassOf :Delay .
78
79  ### urn:mofi-metric#RadioSignalQuality
80  :RadioSignalQuality rdf:type owl:Class ;
81
82                    rdfs:subClassOf mofi:Metric ;
83
84                    rdfs:comment "Represents␣radio␣signal␣quality␣of␣any␣interface␣or␣␣
                          channel."^^xsd:string .
85
86  ### urn:mofi-metric#UsedBandwidth
87  :UsedBandwidth rdf:type owl:Class ;
88
89                rdfs:subClassOf :BandwidthUtilization .
90
91  ### urn:mofi-metric#UsedMemory
92  :UsedMemory rdf:type owl:Class ;
93
94            rdfs:subClassOf :MemoryUtilization ;
95
96             rdfs:comment "Represents␣the␣available␣momory␣that␣is␣unused␣by␣the␣system␣(
                    programs)."^^xsd:string .
```

Listing B.3 shows some classes and properties included in the MOFI Data ontology. The ontology includes even more concepts and relations, which are not represented

in this listing. Similarly, Listing B.4 and Listing B.5 represent the MOFI Tool and Generic Concepts ontologies respectively.

Listing B.3: Part of the MOFI Data ontology

```
@prefix : <urn:mofi-data#> .
...
@base <urn:mofi-data> .

<urn:mofi-data> rdf:type owl:Ontology ;

              rdfs:label "mofi-data"@en .

#################################################################
# Object Properties
#################################################################

### urn:mofi-data#hasMeasurementData
:hasMeasurementData rdf:type owl:ObjectProperty ;

                  rdfs:comment "Different␣data␣which␣can␣be␣obtained␣from␣the␣
                       measurement.␣Those␣Values␣determine␣the␣metrics␣measured␣and␣
                       stated␣in␣:measuresMetric␣property"^^xsd:string ;

                  rdfs:domain mofi:Metric ;

                  rdfs:range :MeasurementData .

### urn:mofi-data#isMeasuredIn

:isMeasuredIn rdf:type owl:ObjectProperty ;

             rdfs:comment "Represents␣the␣unit␣of␣the␣measurement␣data"^^xsd:string ;

             rdfs:domain :MeasurementData ;

             rdfs:range mofi:Unit .

### urn:mofi-data#isMeasurementDataOf

:isMeasurementDataOf rdf:type owl:ObjectProperty ;

                   rdfs:comment "Reprenets␣a␣measurement␣data␣of␣a␣particular␣metric"^^
                        xsd:string ;

                   owl:inverseOf :hasMeasurementData ;

                   rdfs:domain :MeasurementData ;

                   rdfs:range mofi:Metric .

#################################################################
# Data properties
#################################################################

### urn:mofi-data#hasMeasurementDataValue
:hasMeasurementDataValue rdf:type owl:DatatypeProperty ;
```

```
50
51                      rdfs:comment "The value related to a measurement. Should be the
                             property of a MeasurementData (or one of its subclasses)
                             instance and other details as unit and datatype should be
                             present"^^xsd:string ;
52
53                      rdfs:domain [ rdf:type owl:Class ;
54                                    owl:unionOf ( mofi:Metric
55                                                  :MeasurementData
56                                                )
57                                  ] .
58
59 ### urn:mofi-data#hasTimestamp
60 :hasTimestamp rdf:type owl:DatatypeProperty ;
61
62             rdfs:domain :MeasurementData .
63
64 ### urn:mofi-data#hasMeasurementID
65 :hasMeasurementID rdf:type owl:DatatypeProperty ;
66
67                 rdfs:comment "Unique identifier given for each measurement."@en ;
68
69                 rdfs:subPropertyOf omn-lifecycle:hasID ;
70
71                 rdfs:domain mofi:Measurement ;
72
73                 rdfs:range xsd:int .
74
75 ###################################################################
76 # Classes
77 ###################################################################
78
79 ### urn:mofi-data#MeasurementData
80 :MeasurementData rdf:type owl:Class ;
81
82                rdfs:subClassOf mofi:Data ;
83
84                rdfs:comment "Represents individual results of a measured metric."^^xsd:
                       string .
85
86 ### urn:mofi-data#ConfigurationParameter
87 :ConfigurationParameter rdf:type owl:Class ;
88
89                      rdfs:subClassOf mofi:Data ;
90
91                      rdfs:comment "Represents configuration parameters used to to setup
                             the monitoring service, e.g. duration."^^xsd:string .
92
93 ### urn:mofi-data#SimpleMeasurement
94 :SimpleMeasurement rdf:type owl:Class ;
95
96                 rdfs:subClassOf :MeasurementData ;
97
98                 rdfs:comment "Represents data obtained from one simple measurement."^^
                       xsd:string .
99
100 ### urn:mofi-data#StatisticalMeasurement
```

```
:StatisticalMeasurement rdf:type owl:Class ;

                        rdfs:subClassOf :MeasurementData ;

                        rdfs:comment "Represents␣statistical␣measurements␣that␣can␣be␣
                             obtained␣from␣the␣simple␣measurement␣values."^^xsd:string .

### urn:mofi-data#DataFormat
:DataFormat rdf:type owl:Class ;

           rdfs:comment "Represents␣the␣format␣of␣the␣transfered␣data."^^xsd:string .

### urn:mofi-data#JSONFormat
:JSONFormat rdf:type owl:Class ;

           rdfs:subClassOf :FormattedFile ;

           rdfs:comment "Represents␣the␣JSON␣format␣of␣the␣transfered␣data."^^xsd:string
                .

#################################################################
# Individuals
#################################################################

### urn:mofi-data#OMLStream
:OMLStream rdf:type mofi:Data ,
                   :FormattedFile ,
                   owl:NamedIndividual ;

          rdfs:comment "OML␣streams␣exported␣by␣injection␣points␣(OML␣clients)␣and␣
               collected␣by␣the␣OML␣server."^^xsd:string .
```

Listing B.4: Part of the MOFI Tool ontology

```
@prefix : <urn:mofi-tool#> .
...
@base <urn:mofi-tool> .

<urn:mofi-tool> rdf:type owl:Ontology ;

               rdfs:label "mofi-tool"@en .

#################################################################
# Object Properties
#################################################################

### urn:mofi-tool#communicationParadigm
:communicationParadigm rdf:type owl:ObjectProperty ;

                       rdfs:comment "The␣communication␣paradigm␣the␣monitoring␣tool␣is␣
                            using."^^xsd:string ;

                       rdfs:domain :MonitoringTool ;

                       rdfs:range :CommunicationParadigm .

#################################################################
```

```
# Classes
#################################################################

### urn:mofi-tool#MonitoringTool
:MonitoringTool rdf:type owl:Class ;

              rdfs:subClassOf mofi:Tool ;

              rdfs:comment "Represents␣any␣tool␣used␣for␣performing␣monitoring␣service."
                   ^^xsd:string .

### urn:mofi-tool#CaptureTool
:CaptureTool rdf:type owl:Class ;

          rdfs:subClassOf :MonitoringTool ;

          rdfs:comment "Represents␣any␣tool␣used␣for␣capturing␣monitoring␣and␣
               measurement␣data."^^xsd:string .

### urn:mofi-tool#Collector
:Collector rdf:type owl:Class ;

          rdfs:subClassOf :MonitoringTool ;

          rdfs:comment "Represents␣any␣tool␣used␣for␣collecting␣monitoring␣and␣
               measurement␣data."^^xsd:string .

### urn:mofi-tool#Converter
:Adapter rdf:type owl:Class ;

        rdfs:subClassOf :MonitoringTool ;

        rdfs:comment "Represents␣any␣tool␣used␣to␣adapt␣and␣convert␣the␣collected␣data␣
             from␣one␣format␣to␣anohter."^^xsd:string .

## urn:mofi-tool#MeasurementTool
:MeasurementTool rdf:type owl:Class ;

               rdfs:subClassOf mofi:Tool ;

               rdfs:comment "Represents␣any␣tool␣used␣to␣execute␣the␣measurments."^^xsd:
                    string .

## urn:mofi-tool#ActiveMeasurementTool
:ActiveMeasurementTool rdf:type owl:Class ;

                 rdfs:subClassOf :MeasurementTool ;

                 rdfs:comment "Represents␣any␣tool␣used␣to␣execute␣active␣measurments."
                      ^^xsd:string .

## urn:mofi-tool#CommunicationParadigm
:CommunicationParadigm rdf:type owl:Class ;

                    rdfs:comment "Describes␣the␣communication␣paradigm␣used␣in␣the␣tool
                         ␣to␣perform␣the␣measurements␣and␣monitoring␣services."^^xsd:
                         string .
```

```
72
73  ## urn:mofi-tool#ClientServer
74  :ClientServer rdf:type owl:Class ;
75
76              rdfs:subClassOf :Distributed .
77
78  ## urn:mofi-tool#GUI
79  :GUI rdf:type owl:Class ;
80
81      rdfs:subClassOf :DataAccess ;
82
83      rdfs:comment "Represents␣the␣Graphical␣User␣Interface␣used␣to␣get/access␣the␣data."
            ^^xsd:string .
84
85  ## urn:mofi-tool#API
86  :API rdf:type owl:Class ;
87
88      rdfs:subClassOf :DataAccess ;
89
90      rdfs:comment "␣Represents␣the␣Appication␣Programming␣Interface␣used␣to␣get/access␣
            the␣data."^^xsd:string .
91
92  ## urn:mofi-tool#CollectionEndpoint
93  :CollectionEndpoint rdf:type owl:Class ;
94
95                  rdfs:subClassOf omn:Service ;
96
97                  rdfs:comment "␣Represents␣the␣endpoint␣of␣a␣monitoring␣data␣collection
                        ␣service."^^xsd:string .
98
99  ### urn:mofi-tool#Database
100 :Database rdf:type owl:Class ;
101
102         rdfs:subClassOf omn:Service ;
103
104         rdfs:comment "Represents␣any␣database␣management␣system."^^xsd:string .
105
106 ### urn:mofi-tool#SQL
107 :SQL rdf:type owl:Class ;
108
109     rdfs:subClassOf :Database ;
110
111     rdfs:comment "SQL␣(Structured␣Query␣Language)␣is␣a␣special-purpose␣programming␣
            language␣designed␣for␣managing␣data␣held␣in␣a␣relational␣database␣management␣
            system␣(RDBMS),␣or␣for␣stream␣processing␣in␣a␣relational␣data␣stream␣management
            ␣system␣(RDSMS)."^^xsd:string .
112
113 ######################################################################
114 # Individuals
115 ######################################################################
116
117 ### urn:mofi-tool#Ping
118 :Ping rdf:type :CaptureTool ,
119              :MonitoringTool ,
120              :ActiveMeasurementTool ,
121              owl:NamedIndividual ;
122
```

```
123       rdfs:comment "Ping␣is␣a␣program␣that␣can␣be␣considered␣as␣a␣tool␣used␣for␣capturing
              ␣data,␣monitoring␣or␣for␣executing␣active␣measurements.␣The␣ping␣program␣is␣
              can␣be␣used␣to␣test␣wether␣a␣particular␣host␣is␣reachable␣across␣an␣IP␣network.
              ␣It␣measures␣the␣round␣trip␣time,␣packet␣loss,␣and␣some␣statistical␣
              information␣about␣round␣trip␣time␣(minimum,␣maximum,␣average␣and␣mean␣
              deviation)."^^xsd:string .
124
125   ### urn:mofi-tool#Zabbix
126   :Zabbix rdf:type :AnalysisTool ,
127                    :MonitoringTool ,
128                    :GraphicalVisualizationTool ,
129                    owl:NamedIndividual ;
130
131         :communicationParadigm :ClientServer ;
132
133         rdfs:comment "Zabbix␣is␣a␣monitoring␣solution␣used␣to␣monitor␣computer␣and␣
                network␣devices.␣For␣more␣information␣visit␣its␣website:␣http://www.zabbix.
                com."^^xsd:string .
134
135   ### urn:mofi-tool#OMLWrapper
136   :OMLWrapper rdf:type :Adapter ,
137                        owl:NamedIndividual ;
138
139             rdfs:comment "Orbit␣Measurement␣Library␣(OML)␣Wrapper␣(https://oml.mytestbed.
                    net/projects/oml/wiki/)␣is␣a␣software␣that␣acts␣as␣an␣adapter␣retrieving␣
                    measurement␣and␣monitoring␣data␣from␣any␣monitoring␣tool␣and␣converts␣
                    this␣data␣into␣OML␣streams␣that␣are␣then␣sent␣through␣the␣OML␣Measurement
                    ␣Stream␣Protocol␣(OMSP)␣to␣an␣OML␣server."^^xsd:string .
140
141   ### urn:mofi-tool#OMLServer
142   :OMLServer rdf:type omn:Service ,
143                      owl:NamedIndividual ;
144
145             rdfs:comment "OML␣Measurement␣Library␣(OML)␣server␣receives␣measurement␣data␣
                    sent␣by␣OML␣clients␣as␣OML␣streams␣through␣the␣OML␣protocol␣(OMSP)␣and␣
                    stores␣this␣data␣into␣a␣database␣(natively␣support␣SQlite␣and␣PostgreSQL).
                    ␣Website:␣https://oml.mytestbed.net/projects/oml/wiki/."^^xsd:string .
146
147   ### urn:mofi-tool#PostgreSQL
148   :PostgreSQL rdf:type :Collector ,
149                        :SQL ,
150                        owl:NamedIndividual ;
151
152             rdfs:comment "Is␣an␣object-relational␣database␣management␣system.␣Website:␣
                    http://www.postgresql.org."^^xsd:string .
153
154   ### urn:mofi-tool#JenaFuseki
155   :JenaFuseki rdf:type :Collector ,
156                        owl:NamedIndividual ;
157
158             rdfs:comment "Is␣a␣SPARQL␣server␣with␣a␣native␣triple␣store␣database␣(TDB)␣
                    that␣serves␣storing␣and␣querying␣data␣as␣RDF␣triples␣over␣HTTP.␣Website:␣
                    http://jena.apache.org/documentation/serving_data/."^^xsd:string ;
159
160             omn:hasService :SPARQLEndpoint ,
161                            :Database .
162
```

```
### urn:mofi-tool#OMSPEndpoint
:OMSPEndpoint rdf:type :CollectionEndpoint ,
                    owl:NamedIndividual ;

            rdfs:comment "The␣end␣point␣of␣the␣collectin␣service␣that␣offers␣an␣OMSP␣
                interface␣to␣receive␣and␣store␣OML␣measurement␣streams␣transfered␣
                through␣the␣use␣of␣the␣OML␣Measurement␣Stream␣Protocol␣(OMSP)."^^xsd:
                string .

### urn:mofi-tool#SPARQLEndpoint
:SPARQLEndpoint rdf:type :CollectionEndpoint ,
                      owl:NamedIndividual ;

              rdfs:comment "The␣end␣point␣of␣the␣collectin␣service␣that␣offers␣a␣SPARQL␣
                  query␣interface␣allowing␣its␣user␣to␣update␣or␣query␣data␣from␣triple
                  ␣stores␣over␣HTTP␣protocol."^^xsd:string .
```

Listing B.5: Part of the MOFI Generic Concepts ontology

```
@prefix : <mofi-genericconcepts#> .
...
@base <mofi-genericconcepts> .

<urn:mofi-genericconcepts> rdf:type owl:Ontology ;

                        rdfs:label "mofi-genericconcepts"@en .

#################################################################
# Object Properties
#################################################################

### mofi-genericconcepts#locatedAt
:locatedAt rdf:type owl:ObjectProperty ;

          rdfs:range :Location ;

          rdfs:domain [ rdf:type owl:Class ;
                        owl:unionOf ( omn:Service
                                      mofi:Tool
                                      mofi:Measurement
                                      :MonitoringDomain
                                    )
                      ] .

### mofi-genericconcepts#usesProtocol
:usesProtocol rdf:type owl:ObjectProperty ;

            rdfs:range :Protocol ;

            rdfs:domain [ rdf:type owl:Class ;
                          owl:unionOf ( omn:Service
                                        :Protocol
                                      )
                        ] .

### mofi-genericconcepts#query
:query rdf:type owl:ObjectProperty ;
```

```
39
40        rdfs:domain mofi:Tool ;
41
42        rdfs:range [ rdf:type owl:Class ;
43                     owl:unionOf ( omn:Service
44                                   mofi:Tool
45                                 )
46                   ] .
47
48  ######################################################################
49  #
50  # Data properties
51  #
52  ######################################################################
53
54  ### mofi-genericconcepts#latitude
55  :latitude rdf:type owl:DatatypeProperty ;
56
57          rdfs:comment "Determines␣the␣latitude␣coordinate␣of␣a␣physical␣location."^^xsd:
                string ;
58
59          rdfs:domain :PhysicalLocation ;
60
61          rdfs:range xsd:float .
62
63  ### mofi-genericconcepts#longitude
64  :longitude rdf:type owl:DatatypeProperty ;
65
66           rdfs:comment "Determines␣the␣longitude␣coordinate␣of␣a␣physical␣location."^^
                 xsd:string ;
67
68           rdfs:domain :PhysicalLocation ;
69
70           rdfs:range xsd:float .
71
72  ######################################################################
73  # Classes
74  ######################################################################
75
76  ### mofi-genericconcepts#Protocol
77  :Protocol rdf:type owl:Class ;
78
79          rdfs:comment "Represents␣protocols␣used␣for␣measurement␣data␣exchange␣and␣
                reporting."^^xsd:string .
80
81  ### mofi-genericconcepts#Event
82  :Event rdf:type owl:Class ;
83
84        rdfs:comment "Represents␣any␣event␣or␣notification␣within␣a␣system␣during␣
              processing␣phase,␣e.g␣CPU␣load␣is␣over␣a␣defined␣threshold,␣a␣resource␣is␣
              created␣or␣deleted,␣etc.."^^xsd:string .
85
86  ### mofi-genericconcepts#Location
87  :Location rdf:type owl:Class ;
88
```

```
89          rdfs:comment "Represents␣location␣of␣monitoring␣and␣measurement␣related␣
                concepts,␣such␣as␣location␣measurements␣(i.e.␣measured␣metric),␣data␣
                collectors,␣etc."^^xsd:string .
90
91  ### mofi-genericconcepts#MonitoringDomain
92  :MonitoringDomain rdf:type owl:Class ;
93
94                  rdfs:subClassOf omn-federation:Infrastructure ;
95
96                  rdfs:comment "Represents␣the␣domain␣or␣the␣area␣that␣is␣being␣monitored.
                        "^^xsd:string .
97
98  ################################################################
99  # Individuals
100 ################################################################
101
102 ### mofi-genericconcepts#OMSP
103 :OMSP rdf:type :ApplicationProtocol ,
104             owl:NamedIndividual ;
105
106     rdfs:comment "The␣OML␣Measurement␣Stream␣Protocol␣(OMSP)␣is␣used␣by␣the␣Orbit␣
            Measurement␣Library␣(OML)␣to␣describe␣and␣transport␣measurement␣tuples␣between
            ␣injection␣points␣(OML␣clients)␣and␣processing/collection␣points␣(OML␣servers).
            "^^xsd:string .
107
108 ## mofi-genericconcepts#SNMP
109 :SNMP rdf:type :ApplicationProtocol ,
110             owl:NamedIndividual ;
111
112     rdfs:comment "The␣Simple␣Network␣Management␣Protocol␣(SNMP)␣that␣is␣used␣within␣the
            ␣OML␣framework␣for␣transfering␣the␣data␣between␣the␣OML␣clients␣and␣server."^^
            xsd:string .
```

C

Monitoring Resource Adapters

Listing C.1 is an example of an OML wrapper written in Python, which requires the `oml4py.py` and `zabbix_api.py` libraries. This wrapper is one of the MAFIA components that is configured to push monitoring information about PMs hosting users' VMs. It is written to work in threads to support providing data to multiple users at the same time, i.e. it pushes data to different users' collection endpoints (indicated through *collection_uri*) simultaneously. It runs on a regular basis with a frequency of 10 seconds and each time it checks the PMs (referred to with *host_name*), about which monitoring data is required. Such information is stored in an SQLite database (an example from the experiment discussed in Sec. 7.2 is shown in Listing C.2) that is updated by the FITeagle monitoring service module, which is implemented as part of MAFIA. Further configuration information is provided in Listing C.3. The wrapper instantiates communication with the target server (*collection_uri*); see line 77 of Listing C.1. Lines 79 to 81 include the schema that is generated by Scaffold, as discussed in Sec. 6.2.1.4. The wrapper retrieves monitoring data (in this example only three metrics are considered: used memory, used bandwidth and CPU load) from the Zabbix monitoring tool about the PMs using their host names; see lines 110 to 133. The data is then exported as OML streams to the target server as shown in lines 138 to 140.

Listing C.1: Example of an OML wrapper written in Python

```
1  from zabbix_api import ZabbixAPI import oml4py
2  import threading import sqlite3 import sys import logging
3  import logging.handlers import subprocess import pytz
4  import math from datetime import datetime import ast
5  import exceptions import cStringIO import os import re
6  import tempfile import time from time import sleep
7
8  class omlclient (threading.Thread):
9      def __init__(self, threadID, name, targeturi):
10         threading.Thread.__init__(self)
```

```
11          self.threadID = threadID
12          self.name = name
13          self.target = targeturi
14      def run(self):
15          print "Starting " + self.name
16          startOML(self.name, self.target)
17          print self.name + " is terminated."
18
19  def init_logger(settings,name):
20      logger=logging.getLogger(name)
21      logfilename=settings['logger_filename']
22      if(settings['logger_loglevel']=="DEBUG"):
23          loglevel=logging.DEBUG
24      elif settings['logger_loglevel']=="INFO":
25          loglevel=logging.INFO
26      elif settings['logger_loglevel']=="WARNING":
27          loglevel=logging.WARNING
28      else:
29          loglevel=logging.ERROR
30
31      logformatter=logging.Formatter(settings['logger_formatter'])
32      logger.setLevel(loglevel)
33      if(settings['logger_toconsole']=="1"):
34          ch1 = logging.StreamHandler()
35          ch1.setLevel(loglevel)
36          ch1.setFormatter(logformatter)
37          logger.addHandler(ch1)
38      ch2 = logging.handlers.RotatingFileHandler(logfilename, maxBytes=int(settings['
            logger_maxBytes']), backupCount=int(settings['logger_backupCount']))
39      ch2.setLevel(loglevel)
40      ch2.setFormatter(logformatter)
41      logger.addHandler(ch2)
42      return logger
43
44  def read_config(filename):
45      try:
46          f = open(filename, "r")
47      except:
48          logger.error("can not read file %s, script terminated" % (filename))
49          sys.exit()
50      try:
51          dictionsry = {}
52          for line in f:
53              splitchar = '='
54              kv = line.split(splitchar)
55              if (len(kv)==2):
56                  dictionsry[kv[0]] = str(kv[1])[1:-2]
57          return dictionsry
58      except:
59          logger.error("can not read file %s to a dictionary, format must be KEY=VALUE" % (
                filename))
60          sys.exit()
61
62  def checkURL(url):
63      if "http://" not in url :
64          return url
65      else :
```

```
66          return url.replace("http://","")
67
68  def connect_sqlite() :
69      try :
70          con = sqlite3.connect(monitoring_settings['sqliteDB'])
71      except Exception :
72          logger.error("Cannot␣connect␣to␣SQLite3.")
73      return con
74
75  def startOML(threadName, target):
76      #-------------------- schemas as created by Scaffold --------------------#
77      omlInst = oml4py.OMLBase(monitoring_settings["appname"],monitoring_settings["domain"
            ],monitoring_settings["sender"], target)
78
79      omlInst.addmp("used_memory", "used_memory:double:{omn-monitoring-data:
            SimpleMeasurement|omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:
            UsedMemory}{omn-monitoring-metric:UsedMemory|omn-monitoring:
            isMeasurementMetricOf|omn-domain-pc:PC}{omn-monitoring-data:SimpleMeasurement|
            omn-monitoring-data:hasMeasurementDataValue|%value%}{omn-monitoring-data:
            SimpleMeasurement|omn-monitoring:hasUnit|omn-monitoring-unit:Byte}{omn-
            monitoring-unit:Byte|omn-monitoring-unit:hasPrefix|omn-monitoring-unit:giga}␣
            timestamp:datetime:{omn-monitoring-data:SimpleMeasurement|omn-monitoring-data:
            hasTimestamp|%value%}␣physicalresource:string:{omn-domain-pc:PC|omn:hasURI|%
            value%}␣virtualresource:string:{omn-domain-pc:VM|omn:hasURI|%value%}{omn-domain-
            pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}␣")
80      omlInst.addmp("used_bandwidth", "used_bandwidth:double:{omn-monitoring-data:
            SimpleMeasurement|omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:
            UsedBandwidth}{omn-monitoring-metric:UsedBandwidth|omn-monitoring:
            isMeasurementMetricOf|omn-domain-pc:PC}{omn-monitoring-data:SimpleMeasurement|
            omn-monitoring-data:hasMeasurementDataValue|%value%}{omn-monitoring-data:
            SimpleMeasurement|omn-monitoring:hasUnit|omn-monitoring-unit:bitpersecond}{omn-
            monitoring-unit:bitpersecond|omn-monitoring-unit:hasPrefix|omn-monitoring-unit:
            mega}␣timestamp:datetime:{omn-monitoring-data:SimpleMeasurement|omn-monitoring-
            data:hasTimestamp|%value%}␣physicalresource:string:{omn-domain-pc:PC|omn:hasURI
            |%value%}␣virtualresource:string:{omn-domain-pc:VM|omn:hasURI|%value%}{omn-
            domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}␣")
81      omlInst.addmp("cpu_load", "cpu_load:double:{omn-monitoring-data:SimpleMeasurement|omn
            -monitoring-data:isMeasurementDataOf|omn-monitoring-metric:CPULoad}{omn-
            monitoring-metric:CPULoad|omn-monitoring:isMeasurementMetricOf|omn-domain-pc:PC
            }{omn-monitoring-data:SimpleMeasurement|omn-monitoring-data:
            hasMeasurementDataValue|%value%}␣timestamp:datetime:{omn-monitoring-data:
            SimpleMeasurement|omn-monitoring-data:hasTimestamp|%value%}␣physicalresource:
            string:{omn-domain-pc:PC|omn:hasURI|%value%}␣virtualresource:string:{omn-domain-
            pc:VM|omn:hasURI|%value%}{omn-domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:
            PC}␣")
82      #------------------------------------------------------------------------#
83      omlInst.start()
84
85      while True :
86          con = connect_sqlite()
87          logger.debug("%s:␣Connecting␣to␣SQLite..." % threadName)
88
89          with con :
90                 try :
91                     cur = con.cursor()
```

```
92                  cur.execute("select␣distinct(host_name),␣collector_uri,␣vm_uri␣from␣
                        virtual_physical_map␣where␣collector_uri␣like␣\'%" + target + "%\'
                        ")
93                  rows = cur.fetchall()
94                  logger.debug("Fetching␣all␣host␣names␣from␣database...")
95
96              except:
97                  logger.error("Error␣fetching␣data␣from␣SQLite.␣Try␣again␣in␣10␣secs...
                        ")
98
99      if not rows :
100         logger.error("No␣host␣name␣found.␣Exiting␣%s..." % threadName)
101         global listOfCollectorURIs
102         for element in listOfCollectorURIs :
103             if target in element :
104                 listOfCollectorURIs.remove(element)
105                 break
106         break
107
108     else :
109         try:
110             zabbix_server_uri = monitoring_settings['zabbixuri']
111             zapi = ZabbixAPI(server=zabbix_server_uri, log_level=int(
                    monitoring_settings['log_level']))
112             zabbix_username = monitoring_settings['username']
113             zabbix_password = monitoring_settings['password']
114             zapi.login(zabbix_username,zabbix_password)
115
116         except Exception as e:
117             logger.error("can␣not␣open␣zabbix.␣Try␣again␣in␣10␣secs...")
118
119         for row in rows :
120             try:
121                 hostid = zapi.host.get({"filter":{"name":row[0]},"output":"extend"}).
                        pop()['hostid']
122
123                 item = zapi.item.get({"output": "extend","hostids":hostid,"search":{"
                        name":"Used␣memory"}}).pop()
124                 usedmemory = float(item['lastvalue']) / (1024)**3
125                 usedmemory_ts = datetime.fromtimestamp(int(item['lastclock']),pytz.
                        timezone("Europe/Berlin"))
126
127                 item = zapi.item.get({"output": "extend","hostids":hostid,"search":{"
                        key_":"net.if.in[eth2]"}}).pop()
128                 usedbandwidth = float(item['lastvalue']) / (1024)**2
129                 usedbandwidth_ts = datetime.fromtimestamp(int(item['lastclock']),pytz.
                        timezone("Europe/Berlin"))
130
131                 item = zapi.item.get({"output": "extend","hostids":hostid,"search":{"
                        key_":"system.cpu.load[percpu,avg5]"}}).pop()
132                 cpuload = float(item['lastvalue'])
133                 cpuload_ts = datetime.fromtimestamp(int(item['lastclock']),pytz.
                        timezone("Europe/Berlin"))
134
135             except Exception as e:
136                 logger.error("cannot␣fetch␣data␣from␣Zabbix.␣Try␣again␣in␣10␣secs...")
137
```

```
138                 omlInst.inject("used_memory", [usedmemory, usedmemory_ts, row[0], row[2]])
139                 omlInst.inject("used_bandwidth", [usedbandwidth, usedbandwidth_ts, row[0],
                        row[2]])
140                 omlInst.inject("cpu_load", [cpuload, cpuload_ts, row[0], row[2]])
141
142         sleep(10)
143
144     omlInst.close()
145
146 ############################## SCRIPT START ##############################
147 monitoring_settings=read_config('monitoring.cfg')
148 logger=init_logger(monitoring_settings,'monitoring-oml-wrapper.py')
149 logger.debug("monitoring-oml-wrapper.py' has been started")
150 listOfCollectorURIs = []
151 threadCounter = 1
152
153 while True :
154     con = connect_sqlite()
155     logger.debug("MAIN THREAD: Connecting to SQLite...")
156     with con :
157             try :
158                 cur = con.cursor()
159                 cur.execute("select distinct(collector_uri) from virtual_physical_map")
160                 rows = cur.fetchall()
161                 logger.debug("Fetching collector URIs from database...")
162                 for row in rows :
163                     print row[0]
164             except:
165                 logger.error("Error fetching data from SQLite. Try again in 10 secs...")
166
167     if not rows :
168         logger.error("No collector URI found. Try again in 10 secs...")
169
170     else :
171         for row in rows :
172             if row[0] not in listOfCollectorURIs :
173                 uri = checkURL(row[0])
174                 thread_name = "Thread-" + str(threadCounter)
175                 omlclient(threadCounter,thread_name,uri).start()
176                 listOfCollectorURIs.append(row[0])
177                 threadCounter += 1
178                 logger.debug("MAIN THREAD: waiting 10 secs before connecting to SQLite
                        again...")
179
180     sleep(10)
```

Listing C.2: Configuration information stored in SQLite database and required by an OML wrapper for performing its tasks

```
1 sqlite> select * from virtual_physical_map;
2 1041af2d-a25b-48b9-88d0-0cb7f19b1eaf|openstack.av.tu-berlin.de|http
      ://130.149.22.139:3003|http://monitoring.service.tu-berlin.de/resource/Openstack-1/
      c277660a-f5c7-4c29-9fdb-590e6e57ecc2
```

Listing C.3: Configuration information required by an OML wrapper for performing its tasks

```
logger_filename="./monitoring.log"
logger_loglevel="DEBUG"
logger_maxBytes="10485760"
logger_backupCount="2"
logger_toconsole="1"
logger_formatter="%(asctime)s␣-␣%(name)s␣-␣%(levelname)s␣-␣%(message)s"

#SQLite database path + name
sqliteDB="monitoring_sqliteDB.db"

#oml app name
appname="infrastructure_monitoring"
#oml domain name
domain="experiment_demo"
#oml sender
sender="http://monitoring.service.tu-berlin.de"

#zabbix uri
zabbixuri="http://<server_IP>/zabbix"
#zabbix username
username="<username>"
#zabbix password
password="<password>"
#zabbix API log level
log_level="0"
```

Listing C.4 is another example of an OML wrapper written in Ruby. It requires the `oml4rb` and `zabbixapi` libraries. This wrapper represents a simple case where monitoring data about the used memory, used bandwidth and CPU load of a PM (identified in line 94) hosting a user VM (identified in line 95) is provided regularly every 30 seconds. It gets data from Zabbix and provides it as OML streams to the target server whose URI is read from a configuration file (see line 78) along with further configuration information like that identified in Listing C.3.

Listing C.4: An example of OML wrapper written in Ruby

```
require "oml4r"
require "zabbixapi"
require 'date'
require 'time'

#----------------------- Schemas as screated by Scaffold -----------------------#

class USED_MEMORY_MP < OML4R::MPBase
 name :used_memory
 param :used_memory, {:type => :double, :relation => "{omn-monitoring-data:
      SimpleMeasurement|omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:
      UsedMemory}{omn-monitoring-metric:UsedMemory|omn-monitoring:isMeasurementMetricOf|
      omn-domain-pc:PC}{omn-monitoring-data:SimpleMeasurement|omn-monitoring-data:
      hasMeasurementDataValue|%value%}{omn-monitoring-data:SimpleMeasurement|omn-
      monitoring:hasUnit|omn-monitoring-unit:Byte}{omn-monitoring-unit:Byte|omn-
      monitoring-unit:hasPrefix|omn-monitoring-unit:giga}"}
```

```
11  param :timestamp, {:type => :datetime, :relation => "{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring-data:hasTimestamp|%value%}"}
12  param :physicalresource, {:type => :string, :relation => "{omn-domain-pc:PC|omn:hasURI
        |%value%}"}
13  param :virtualresource, {:type => :string, :relation => "{omn-domain-pc:VM|omn:hasURI|%
        value%}{omn-domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}"}
14 end
15
16 class USED_BANDWIDTH_MP < OML4R::MPBase
17  name :used_bandwidth
18  param :used_bandwidth, {:type => :double, :relation => "{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:
        UsedBandwidth}{omn-monitoring-metric:UsedBandwidth|omn-monitoring:
        isMeasurementMetricOf|omn-domain-pc:PC}{omn-monitoring-data:SimpleMeasurement|omn-
        monitoring-data:hasMeasurementDataValue|%value%}{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring:hasUnit|omn-monitoring-unit:bitpersecond}{omn-
        monitoring-unit:bitpersecond|omn-monitoring-unit:hasPrefix|omn-monitoring-unit:mega
        }"}
19  param :timestamp, {:type => :datetime, :relation => "{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring-data:hasTimestamp|%value%}"}
20  param :physicalresource, {:type => :string, :relation => "{omn-domain-pc:PC|omn:hasURI
        |%value%}"}
21  param :virtualresource, {:type => :string, :relation => "{omn-domain-pc:VM|omn:hasURI|%
        value%}{omn-domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}"}
22 end
23
24 class CPU_LOAD_MP < OML4R::MPBase
25  name :cpu_load
26  param :cpu_load, {:type => :double, :relation => "{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring-data:isMeasurementDataOf|omn-monitoring-metric:
        CPULoad}{omn-monitoring-metric:CPULoad|omn-monitoring:isMeasurementMetricOf|omn-
        domain-pc:PC}{omn-monitoring-data:SimpleMeasurement|omn-monitoring-data:
        hasMeasurementDataValue|%value%}"}
27  param :timestamp, {:type => :datetime, :relation => "{omn-monitoring-data:
        SimpleMeasurement|omn-monitoring-data:hasTimestamp|%value%}"}
28  param :physicalresource, {:type => :string, :relation => "{omn-domain-pc:PC|omn:hasURI
        |%value%}"}
29  param :virtualresource, {:type => :string, :relation => "{omn-domain-pc:VM|omn:hasURI|%
        value%}{omn-domain-pc:VM|omn-lifecycle:childOf|omn-domain-pc:PC}"}
30 end
31
32 #------------------------------------------------------------------------#
33
34 #write code here
35 def get_vars(conf_file)
36     #Here a couple of things are defined
37     #First is the regular expression that is used to
38     #get rid of whitespace and the array characters.
39     line_sub = Regexp.new(/\s+|\\|"|\[|\]/)
40
41 ␣␣␣␣temp␣=␣Array.new
42 ␣␣␣␣vars=Hash.new
43
44 ␣␣␣␣#Check␣and␣make␣sure␣that␣the␣file␣exists
45 ␣␣␣␣unless␣File.exists?(conf_file)␣then
46 ␣␣␣␣␣␣raise␣"The specified configuration file doesn't␣exist!"␣
47 ␣␣␣␣end
```

```
48  ␣␣␣␣IO.foreach(conf_file)␣do␣|line|
49  ␣␣␣␣␣␣#discard␣comment␣lines
50  ␣␣␣␣␣␣if␣line.match(/^#/)
51  ␣␣␣␣␣␣␣␣next
52  ␣␣␣␣␣␣elsif
53  ␣␣␣␣␣␣␣␣#discard␣a␣blank␣line
54  ␣␣␣␣␣␣␣␣line.match(/^$/)
55  ␣␣␣␣␣␣␣␣next
56  ␣␣␣␣␣␣else
57  ␣␣␣␣␣␣␣␣#Snag␣variable␣and␣throw␣it␣into␣the␣varhash
58  ␣␣␣␣␣␣␣␣temp[0],temp[1]␣=␣line.to_s.scan(/^.*$/).to_s.split('=')
59
60  ␣␣␣␣␣␣␣␣#Match␣our␣regular␣expression␣and␣substitute
61  ␣␣␣␣␣␣␣␣temp.collect!␣do␣|val|
62  ␣␣␣␣␣␣␣␣␣␣val.gsub(line_sub,␣"")
63  ␣␣␣␣␣␣␣␣end
64  ␣␣␣␣␣␣␣␣#Add␣the␣variables␣to␣our␣hash
65  ␣␣␣␣␣␣␣␣vars[temp[0]]␣=␣temp[1]
66  ␣␣␣␣␣␣end
67  ␣␣␣␣end
68  ␣␣␣␣#And␣return␣them
69  ␣␣␣␣return␣vars
70  ␣␣end
71
72  conf␣=␣get_vars("../monitoring.cfg")
73
74  oml_opts␣=␣{
75  ␣␣:appName␣=>␣conf['appname'],
76  ␣␣:domain␣=>␣conf['domain'],
77  ␣␣:nodeID␣=>␣conf['sender'],
78  ␣␣:collect␣=>␣conf['target']
79  }
80
81  zabbix_opts␣=␣{
82  ␣␣:url␣=>␣conf['zabbixuri']␣+␣'/api_jsonrpc.php',
83  ␣␣:user␣=>␣conf['username'],
84  ␣␣:password␣=>␣conf['password']
85  }
86
87  OML4R::init(ARGV,␣oml_opts)
88
89  while␣true
90
91  ␣␣zbx␣=␣ZabbixApi.connect(zabbix_opts)
92
93  ␣␣begin
94  ␣␣␣␣␣␣hostname␣=␣"<hostname>"
95  ␣␣␣␣␣␣vm␣=␣"<vm>"
96
97  ␣␣␣␣host␣=␣zbx.query(
98  ␣␣␣␣␣␣:method␣=>␣"host.get",
99  ␣␣␣␣␣␣:params␣=>␣{
100 ␣␣␣␣␣␣␣␣:output␣=>␣"extend",
101 ␣␣␣␣␣␣␣␣:search␣=>␣{
102 ␣␣␣␣␣␣␣␣␣␣:name␣=>␣hostname
103 ␣␣␣␣␣␣␣␣}
104 ␣␣␣␣␣␣}
```

```
105 ␣␣␣␣)
106 ␣␣␣␣unless␣host.empty?
107 ␣␣␣␣␣␣hostid␣=␣host[0]["hostid"]
108 ␣␣␣␣end
109
110 ␣␣␣␣used_memory␣=␣zbx.query(
111 ␣␣␣␣␣␣:method␣=>␣"item.get",
112 ␣␣␣␣␣␣:params␣=>␣{
113 ␣␣␣␣␣␣␣␣:output␣=>␣"extend",
114 ␣␣␣␣␣␣␣␣:hostids␣=>␣hostid,
115 ␣␣␣␣␣␣␣␣:search␣=>␣{
116 ␣␣␣␣␣␣␣␣␣␣:name␣=>␣"Used␣memory"
117 ␣␣␣␣␣␣␣␣}
118 ␣␣␣␣␣␣}
119 ␣␣␣␣)
120 ␣␣␣␣unless␣used_memory.empty?
121 ␣␣␣␣␣␣used_memory_ts␣=␣used_memory[0]["lastclock"].to_i
122 ␣␣␣␣␣␣used_memory␣=␣used_memory[0]["lastvalue"].to_f␣/␣(1024)**3
123 ␣␣␣␣␣␣USED_MEMORY_MP.inject(used_memory,␣Time.at(used_memory_ts).localtime.strftime("%Y
      -%m-%dT%H:%M:%S"),␣hostname,␣vm)
124 ␣␣␣␣end
125
126 ␣␣␣␣used_bandwidth␣=␣zbx.query(
127 ␣␣␣␣␣␣:method␣=>␣"item.get",
128 ␣␣␣␣␣␣:params␣=>␣{
129 ␣␣␣␣␣␣␣␣:output␣=>␣"extend",
130 ␣␣␣␣␣␣␣␣:hostids␣=>␣hostid,
131 ␣␣␣␣␣␣␣␣:search␣=>␣{
132 ␣␣␣␣␣␣␣␣␣␣:key_␣=>␣"net.if.in[eth2]"
133 ␣␣␣␣␣␣␣␣}
134 ␣␣␣␣␣␣}
135 ␣␣␣␣)
136 ␣␣␣␣unless␣used_bandwidth.empty?
137 ␣␣␣␣␣␣used_bandwidth_ts␣=␣used_bandwidth[0]["lastclock"].to_i
138 ␣␣␣␣␣␣used_bandwidth␣=␣used_bandwidth[0]["lastvalue"].to_f␣/␣(1024)**2
139 ␣␣␣␣␣␣USED_BANDWIDTH_MP.inject(used_bandwidth,␣Time.at(used_bandwidth_ts).localtime.
      strftime("%Y-%m-%dT%H:%M:%S"),␣hostname,␣vm)
140 ␣␣␣␣end
141 ␣␣
142 ␣␣end
143 ␣␣sleep␣30
144 end
145
146 OML4R::close()
```

D

Evaluation Appendix

This section gives further details on the experiments conducted in the evaluations discussed in Chapter 7.

The SPARQL queries performed in the experiment discussed in Sec. 7.2 are listed in Listing D.1, Listing D.2 and Listing D.3.

Listing D.1: SPARQL query to get delay of a link

```
SELECT ?timevalue ?value ?prefix ?unit{{
?measure omn-monitoring-data:isMeasurementDataOf ?metric .
?metric rdf:type omn-monitoring-metric:Delay .
?metric omn-monitoring:isMeasurementMetricOf ?resource .
?measure omn-monitoring-data:hasMeasurementDataValue ?value .
?measure omn-monitoring:hasUnit ?unit_value .
?unit_value rdf:type ?unit .
?unit_value omn-monitoring-unit:hasPrefix ?prefix_value .
?prefix_value rdf:type ?prefix .
?measure omn-monitoring-data:hasTimestamp ?timevalue .
?resource omn:hasURI ?uri .
filter(regex(?uri,"http://localhost/Link_1"))
}}
```

Listing D.2: SPARQL query to get bandwidth of a particular PM resource

```
SELECT ?timevalue ?value ?prefix ?unit{{
?measure omn-monitoring-data:isMeasurementDataOf ?metric .
?metric rdf:type omn-monitoring-metric:UsedBandwidth .
?metric omn-monitoring:isMeasurementMetricOf ?resource .
?measure omn-monitoring-data:hasMeasurementDataValue ?value .
?measure omn-monitoring:hasUnit ?unit_value .
?unit_value rdf:type ?unit .
?unit_value omn-monitoring-unit:hasPrefix ?prefix_value .
?prefix_value rdf:type ?prefix .
?measure omn-monitoring-data:hasTimestamp ?timevalue .
?resource omn:hasURI ?uri .
filter(regex(?uri,"openstack.av.tu-berlin.de"))
}}
```

Listing D.3: SPARQL query to get bandwidth of a particular VM resource

```
SELECT ?timevalue ?value ?prefix ?unit{{
?measure omn-monitoring-data:isMeasurementDataOf ?metric .
?metric rdf:type omn-monitoring-metric:UsedBandwidth .
?metric omn-monitoring:isMeasurementMetricOf ?resource .
?measure omn-monitoring-data:hasMeasurementDataValue ?value .
?measure omn-monitoring:hasUnit ?unit_value .
?unit_value rdf:type ?unit .
?unit_value omn-monitoring-unit:hasPrefix ?prefix_value .
?prefix_value rdf:type ?prefix .
?measure omn-monitoring-data:hasTimestamp ?timevalue .
?resource omn:hasURI ?uri .
filter(regex(?uri,"http://monitoring.service.tu-berlin.de/resource/Openstack-1/c277660a-
    f5c7-4c29-9fdb-590e6e57ecc2"))
}}
```

Figure D.2 shows an extended an RDF-based graph, which represents information related to the experiment conducted and discussed in Sec. 7.2. It is an extended graph of the graph shown in Figure 7.10

Figure D.1 shows an example of how the queueing delay in an OML server increases the $\boldsymbol{T_{delay}}$, defined and discussed in Sec. 7.3.3. This figure illustrates a sequence diagram indicating how the $\boldsymbol{T_{delay}}$ is increased due to the queuing delays at the server in the case of a batch of five succesive OML streams (representing five measurement metrics).

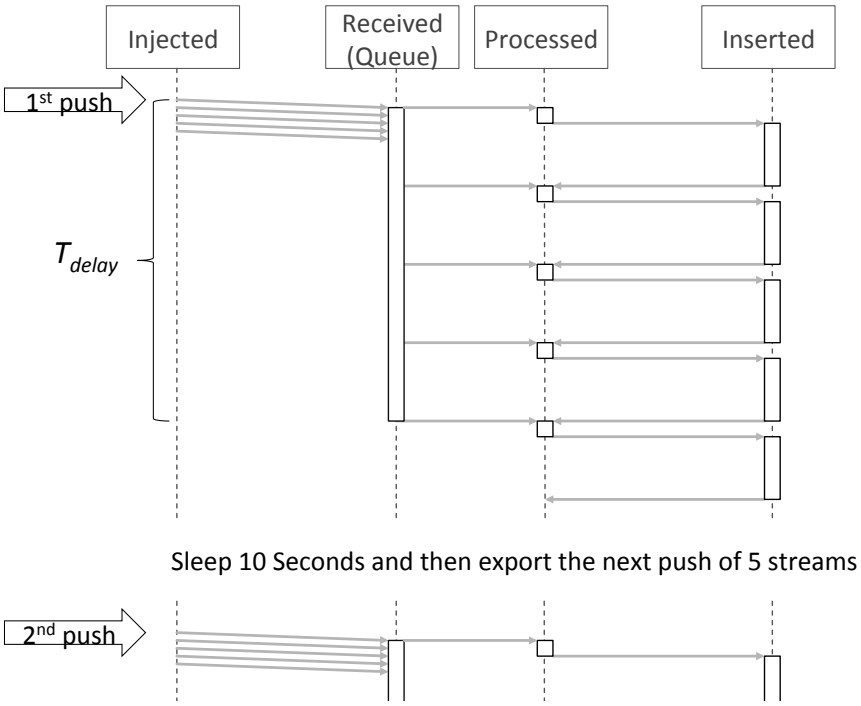


Fig. D.1.: A sequence diagram indicating processing at an OML server of a batch of 5 successive OML streams

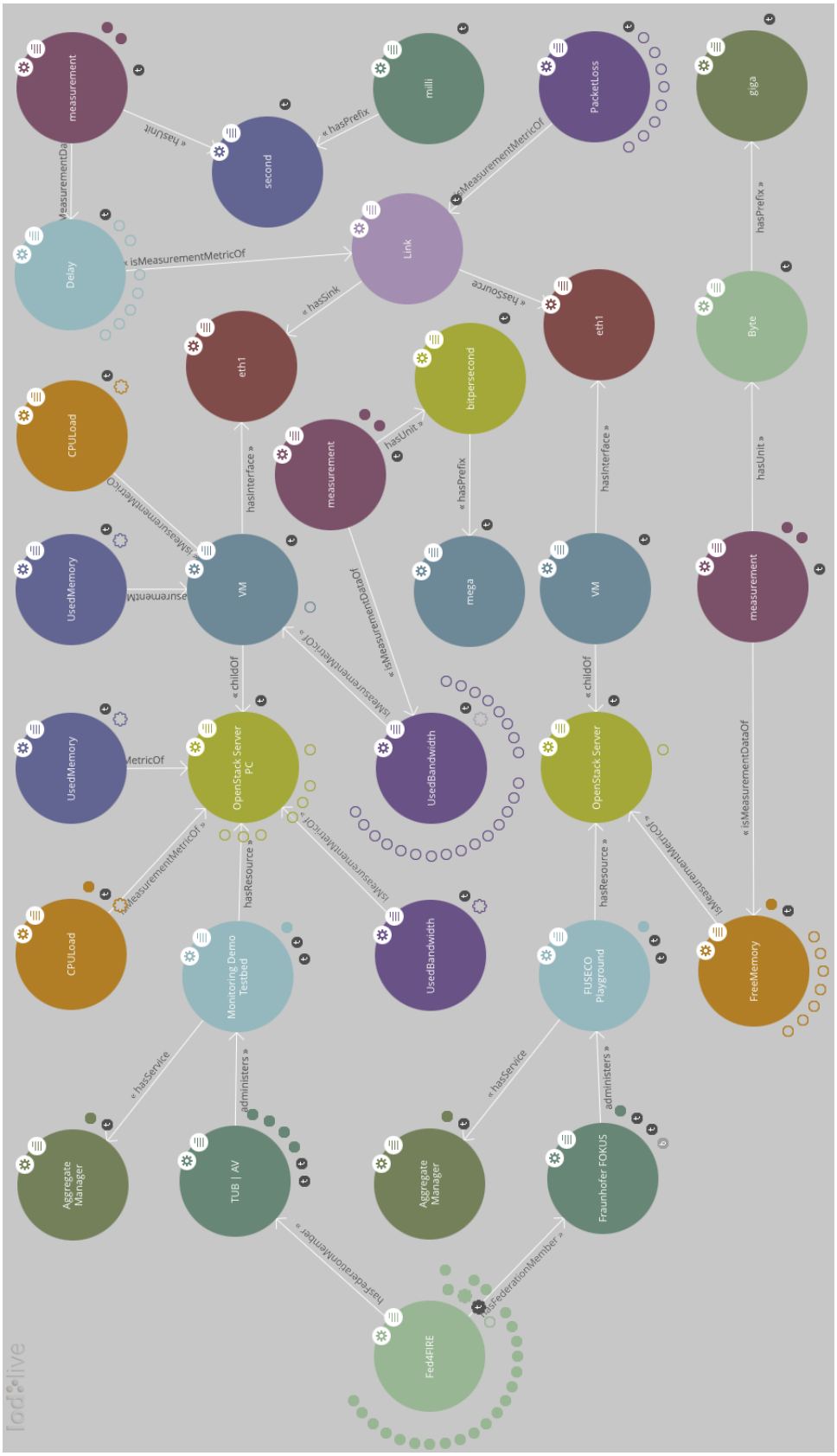


Fig. D.2.: RDF graph representation in Lodlive

E

Glossary

This section explains the relevant technical terms and definitions used most frequently throughout this thesis.

Cloud Computing – It is defined by NIST as follows: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

Cloud Infrastructure – It is defined by NIST as follows: "A cloud infrastructure is the collection of hardware and software that enables the five essential characteristics of cloud computing. The cloud infrastructure can be viewed as containing both a physical layer and an abstraction layer. The physical layer consists of the hardware resources that are necessary to support the cloud services being provided, and typically includes server, storage and network components. The abstraction layer consists of the software deployed across the physical layer, which manifests the essential cloud characteristics. Conceptually the abstraction layer sits above the physical layer."

Cross-domain monitoring – Monitoring various components, systems, network and software metrics that are distributed across multiple domains.

Cross-layer Monitoring – Monitoring all functional layers expanding from low-level resources (e.g. CPU, memory) through the network up to the services and applications layers.

Data Model – As defined by NOVI, a data model "describes protocols and implementation details, based on the representation of concepts and their relations provided by the information model".

Experimenters – Experimenter are researchers from academia and industry, or developers who implement and test novel applications, technologies or protocols.

Federation – "A model for the establishment of a large scale and diverse infrastructure for the communication technologies, services, and applications and can generally be seen as an interconnection of two or more independent administrative domains for the creation of a richer environment and for the increased multilateral benefits of the users of the individual domains".[9]

Federation Services – Those services deployed by the federation operators or administrators to ensure both the health of the individual infrastructures participating in the federation, and the performance and the quality of their offerings. The main federation services relevant to the federation use cases addressed in this thesis are SLA management, trustworthy reputation, reservation, brokerage, and FLS monitoring dashboard.

Future Internet – The growing success and the increasing confidence in the Internet in our daily lives have led to major debate among experts on the ability of the current architecture to be repaired. If not, it may collapse under the increasing demand of future applications [5], [65]. David D. Clark, MIT, in an article in *MIT Technology Review* in 2005 said "The Internet is broken". As a matter of fact, several global activities propose FI architectures (following either clean-slate or evolutionary Internet design approaches) to solve the limitations [4], [5], [66] of the current Internet. The Future Internet has multiple pillars, among these are: i) Cloud Computing, ii) Network of the Future (NoF) focusing on OpenFlow/SDN, network virtualization by decoupling networks from infrastructures and providing isolated VNs with different properties or even individual VNFs, iii) IoT, iv) IoS, and v) Internet of Content (IoC).

Information Model – It is a conceptual model for designers to describe entities and their relationships within a system at a conceptual level.

Measurement – As defined by Weiner [87] measurement is: "A systematic, replicable process by which objects or events are quantified and/or classified with respect to a particular dimension. This is usually achieved by the assignment of numerical values."

Monitoring – The process of constantly observing and recording information about resources (virtual and physical devices, systems, processes, applications, networks, traffic flow, etc.) to determine their state, usage, performance, and

behavior. This information is used by various users and systems that are responsible for controlling and managing these resources, as well as further services such as capacity planning, SLA management, trustworthy reputation, security and privacy assurance, data analytics, etc.

Monitoring Information – It refers to any piece of data that carries state information of a monitored entity.

Monitoring Ontology – It is a formal representation of resources, resource-specific metrics and their measured data as a set of concepts, and the relationships between those concepts.

Ontology – As defined in [155], "An Ontology is a formal, explicit specification of a shared conceptualization". An ontology defines a set of formal, explicit vocabularies and definitions of concepts and their relationships within a given domain.

Testbed – "An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test".[7]

Users – The term 'user' is used in this thesis to refer to a service provider or application developer in a commercial cloud or any similar federated ICT infrastructure, or to an experimenter in a cloud-based or FI testbed. Traditional end-users who are using the Internet and telecom services and applications are excluded.

User-Customized Resource Environment (UCRE) – It is a collection of resources (compute, network, storage, software, etc.) that are created on-demand, configured and used by a user or even a group of users with the correct permissions. When referring to a UCRE in this thesis, we are referring to a cloud service (e.g. PaaS or IaaS) in cloud infrastructures or to an experiment in cloud-based or FI testbeds.