Objective FP7-ICT-2009-5-257448/D-2.2

Future Networks

Project 257448

"SAIL – Scalable and Adaptable Internet Solutions"

D-2.2
(D-A.2) Draft Architectural Guidelines and Principles

Date of preparation: **11-07-31**
Start date of Project: **10-08-01**
Project Coordinator: **Thomas Edwall**
Ericsson AB

Revision: **1.0**
Duration: **13-01-31**

Document Properties

Document Number:	D-2.2
Document Title:	**(D-A.2) Draft Architecture Guidelines and Principles**
Document Responsible:	Benoit Tremblay (EAB)
Document Editor:	Benoit Tremblay (EAB), Peter Schoo (Fraunhofer)
Authors:	Pedro Aranda (TID) Jorge Carapinha (PTIN) Dominique Dudkowski (NEC) Peter Schoo (Fraunhofer) Michael Soellner (ALUD) Benoit Tremblay (EAB)
Target Dissemination Level:	PU
Status of the Document:	Final Version
Version:	1.0

Production Properties:

Reviewers:	Holger Karl (UPB) Michael Soellner (ALUD) Luis M. Correia (IST)

Document History:

Revision	Date	Issued by	Description
1.0	2011-07-29	Peter Schoo	Final Version

Disclaimer:

This document has been produced in the context of the SAIL Project. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2010–2013) under grant agreement n° 257448.

Abstract:

This document presents principles and guidelines applicable to the architecture work being done in SAIL. Apart from generic principles and guidelines, objectives of the Themes are presented as well as the approach taken to reach those objectives.
The document includes a simplified view of the SAIL overall architecture, identifying the main interfaces between the work packages and preliminary results from the Theme work. This architecture will be refined as the architecture of each work package will evolved.
Finally, an evaluation of the work done so far and identification of open issues are presented in the concluding chapter.

Keywords:

Architecture, Future Internet, SAIL, Information-Centric, Networking, Cloud Computing

Executive Summary

This document is a public deliverable of the Scalable Adaptive Internet Solution (SAIL) EU-FP7 project [1] and describes the architectural principles and guidelines to be followed during the project.

SAIL work is divided in four work package (WP). Out of these four, three are carrying out architectural work: Network of Information (NetInf, WP-B), Open Connectivity Services (OConS, WP-C) and Cloud Networking (CloNe, WP-D). These three work packages target different aspects of the Future Internet and have different perspectives on the functions to be implemented. To help interested readers, both external and internal to the project, to have a global understanding, this document establishes the base to harmonise the architecture work in the project.

First, a set of common terms that are used throughout the project is established.

A set of generic principles and guidelines is put in place to orient the design along with some architectural guidelines and objectives for each of the Themes defined for the project (Inter-Provider, Network Management, Prototyping & Experimentation and Security). Since SAIL is targeting deployment in a three to five years time frame, migration considerations have also been added.

A list of recommended topics to be covered by the WP architectural documents is enumerated.

A simplified overview of the SAIL global architecture is presented where the main interfaces between the different systems studied in SAIL are identified. Some early results from the Theme work are also presented.

Finally, an evaluation of the work done so far and identification of open issues are presented in the concluding chapter.

Contents

List of Figures

List of Acronyms

API Application Programing Interface

AS Autonomous System

CIA Confidentiality, Integrity and Availability

CloNe Cloud Networking

DFZ Default Free Zone

DE Decision Making Entity

DFM Distributed Fault Management

DGT Distributed Goal Translation

DRM Distributed Resource Management

DTN Delay Tolerant Network

EE Execution and Enforcement Entity

FNS Flash Network Slice

FOSS Free and Open Source Software

FSF Free Software Foundation

GPL GNU General Public License

IE Information Management Entity

INM In-Network Management

IO Information Object

IP Integrated Project

IPR Intellectual Property Rights

MC Management Capability

MDHT Multiple Distributed Hash Table

MVNO Mobile Virtual Network Operator

NC NetInf Cache

NR NetInf Router

NRS Name Resolution System

NTS NetInf Transport Service

OConS Open Connectivity Services

OF Orchestration Function

OSI Open Source Initiative

PKI Public Key Infrastructure

PMT Project Management Team

REST Representational State Transfer

SAIL Scalable Adaptive Internet Solution

SAP Service Access Point

SE Self-Managing Entity

TOGAF The Open Group Architecture Framework

UNI User-Network Interface

VPN Virtual Private Network

WP work package

1 Introduction

1.1 Motivation and objectives

A lot of architectural work is ongoing in the Scalable Adaptive Internet Solution (SAIL) project. Each work package (WP) has two deliverables related to architecture. At first, an initial architecture is presented by each WP at the end of the first year of the project. Based on these preliminary architectures, experimentation will be done. Improvement to issues observed during the experimentation phase will be integrated by a second round of architecture documents, occurring closer to the end of the project. Finally, at the end of the project, a harmonised architecture integrating the views from the work packages and the Themes will be published.

Each WP tackles a different set of functions and applications and thus has different perspectives on the network. Despite the fact that the architectural elements to be described are not the same, there are some principles that are applicable to all of them. Also, in SAIL, we have defined four cross-cutting themes to ensure cohesion between the work packages in key areas. These four Themes are Security, Network Management, Inter-Provider, and Experimentation & Prototyping. Each of those Themes have established guidelines to be addressed by the work packages. Since SAIL is targeting a deployment in 3 to 5 years, migration guidelines are presented in addition to the four Themes.

The objectives of the current document are to establish a common understanding of the architectural work, to introduce a common terminology, to present architecture principles, and to provide guidelines on how aspects common to all work packages should be addressed. The document also aims at setting the base of the integrated overall SAIL architecture.

As the guidelines are established in parallel with the ongoing architecture work in each work package, the initial architecture documents might not be totally in line with the guidelines expressed in the current document. However, we expect that those guidelines will be followed for the final releases of the architecture descriptions.

1.2 Structure

Apart from the current introduction, the core of the document is divided in five chapters. The next chapter introduces some *terminology* used in SAIL. In Chapter 3, we introduce some *generic business and technological principles*. We also provide *guidelines* and *framework for each of the Themes* covered by SAIL and also for migration. In Chapter 4, we enumerate a list of *topics that should be covered by the architecture documents*. We introduce a preliminary and simplified version of the *overall architecture* and identify the major interfaces in Chapter 5 along with some *preliminary results* from the Themes. Finally, in Chapter 6, we summarise the work done, present a first evaluation of the architectural work and identify open issues.

2 Glossary

This chapter introduces a list of common terms that are used throughout the many deliverables of the SAIL project. This list does not pretend to be exhaustive.

- **Access Provider** The party that provides, authorises, and manages the immediate physical access of the end user. It could be a fixed access provider (e.g. DSL), a mobile/wireless access provider (e.g. 3G, WLAN), or an enterprise for its corporate network.
- **Application Programing Interface (API)** An Application Programing Interface is the description of the functions, data structure and rules or usage patterns to be used by a programmer on either side of an interface. An API can be mapped to one or more programming languages (language binding). The pragmatic aspects of an API relate to the underlying model of the supported interactions that an API enables (usage patterns, call sequences) and the underlying model of interface binding (e.g. Representational State Transfer, RESTful).
- **Connectivity Resources** Physical or virtual nodes and links used to established connectivity between end-points. Those end-points can be hosts, data centres, or network interconnects.
- **Demonstration** A demonstration is a scientific/technical experiment carried out for the purposes of proving scientific/technical effects or solutions, rather than for hypothesis testing or knowledge gathering (although it may originally have been carried out for these purposes).
- **End User** The end user is the source or destination party that initiates (connects) or receives/terminates requests towards the network (i.e. other stakeholders) – it could be a human, a machine, an abstract higher-layer application, or an information object (content).
- **Experimentation** Experimentation is the step in the scientific method that arbitrates between competing models or hypotheses. Experimentation is also used to test existing theories or new hypotheses in order to support them or disprove them. Moreover, an experiment may also test a question or test previous results.
- **Flash network slice** A network resource that can be provisioned and dimensioned on a time scale comparable to existing compute and storage resources. Flash network slices can be used to construct and connect virtual infrastructures spanning providers.
- **Free and Open Source Software (FOSS)** Also free/libre/open-source software (**FLOSS**) – Liberally licensed software to grant the right of users to use, study, change, and improve its design through the availability of its source code.
- **Guideline** *A statement or other indication of policy or procedure by which to determine a course of action.* [5] Following a guideline is not usually considered to be mandatory.
- **Innovation** Innovation generally refers to the creation or (substantial) improvement of products, technologies, or ideas. The goal of innovation is positive change, to make someone or something significantly better, e.g. increase value, customer value, or producer value. The literature on innovation typically distinguishes between invention, an idea made manifest, and innovation, ideas applied successfully in practice.

- **Interface** An interface is the point of interaction between two systems. Its description usually includes the information exchanged, the rules on how that information can be exchanged and the expected behaviour of each system after receiving the information. The interface may be an programmatic interface, like an API that can hide implementation details; it may be an interworking interface, like a protocol that allows communication by exchange of messages between two or more systems.

- **Network/infrastructure/resource provider** The party that owns and manages the network infrastructure resources or parts of it. Multiple network/infrastructure providers may be connected horizontally (on a peer-to-peer model) or vertically (forming a hierarchy of service providers).

- **Network Domain** The concept *network domain* can be very different depending on the aspects that need to be highlighted. In the case of SAIL, three aspects are investigated: the implications of **administrative, operational, and trust domains** and their borders. The Inter-provider Theme concentrates mainly on administrative and operational domains. Trust domains are in the scope of the Security Theme.
  - **Administrative Domain** An **Administrative Domain** is a collection of network resources that are under the control of a single administrator. This definition is shared by all work packages. WP-D refers to it simply as **Network Domain**.
  - **Operational Domain** **Operational Domains** are collections of nodes that share the same operational procedures. Examples of operational domains are:
    1. An **administrative domain** that can be partitioned into its legacy Internet operation and an ICN network would have two operational domains.
    2. a Name Resolution Region's hierarchy that can be addressed through a Multiple Distributed Hash Table (MDHT) Name Resolution System (NRS)
    3. regions where, due to lack of connectivity, a common NRS cannot be used, like e.g. Delay Tolerant Networks (DTNs), would represent different **Operational Domains**.
  - **Trust Domain** A **trust domain** consists of a group of people, information resources, data systems and/or networks that share a set of rules governing access to data and services. This set of rules is known as a *security policy*.

- **Principle** Out of the many definitions for Principle, we embrace the following definitions. *A principle can be a fundamental law or a law of nature that can be observed as a natural phenomenon or the behaviour of a system.* On the other hand, *a principle is also a moral law set to orientate the conduct. It establishes an obligation to follow that law.* [6, 7] In both cases a principle is understood as rule without exception.
  Principles are general rules intended to be enduring and seldom amended, that inform and support the way in which an organisation (here: the SAIL project) sets its vision and values about fulfilling its mission. Architecture principles are fundamental to making decisions related to the design and construction of the architected system [8].

- **Proof-of-Concept (PoC)** Proof-of-Concept is a realisation of a certain method or idea(s) to demonstrate its feasibility, or a demonstration in principle, whose purpose is to verify that some concept or theory is probably capable of being useful. A proof-of-concept may or may not be complete, and is usually small and incomplete.

- **Protocol** A protocol is the formal description of the messages and rules for exchanging those messages between two systems. Such a communication protocol defines the syntax and

semantics and encompasses the pragmatic aspects of the underlying model of the supported interactions (cf. RESTful).

- **Prototype** A Prototype is a first version of a product or system component meant for demonstration, experimentation and testing purposes only. A prototype typically simulates only a few aspects of the features of the eventual system. Basic prototype categories are proof-of-concept prototype, functional prototype, or visual prototype.

- **REST** Representational State Transfer (REST) is a style of software architecture for distributed systems. [9]

- **SAP** A Service Access Point (SAP) is the point of interaction provided by a lower layer to the next layer above in a multi-layer protocol stack (or to other layers if cross-layer interactions are allowed by the architecture). SAPs formalise the notion of an interface.

- **Scenario** Scenario is a wider application area where a particular technology can be useful. A scenario includes multiple use cases.

- **Security objectives** Such objectives describe a protection target according to some security policy (see Figure 2.1). Security services like authentication, confidentiality, or integrity service are used to implement security policies. As such, the security objectives determine which security services need to be used.

- **Security management** Management process to maintain the intended protection level after a system was released. Often security does not last forever, because of e.g. mathematical advances that improve cryptographic analysis and thus endanger properties of a security service implementation or because of newly identified vulnerabilities in the implementation. Usual security management involves patch management as an instrument.

- **Security services** Characterise the provided security properties. For example, a complete set could be authentication, authorisation, confidentiality, integrity, or non-repudiation. There are also other classical models like Confidentiality, Integrity and Availability (CIA). The explicit enumeration of security services determines the exact and implementation-independent security properties.

- **System Architecture, Framework** The structure of system components, their relationships, and the principles and guidelines governing their design and evolution over time [10, 11]. A framework provides partial solutions or even half-products, like for example dedicated code libraries, to build a system according to an architecture.

- **User-Network Interface (UNI)** A User-Network Interface (UNI) is a specific type of interface between a client and a transport network to allow the client to establish and control connectivity. Example of UNI can be found in [12].

- **Use Case** A use case describes how a particular technology can be used to solve a problem or satisfy a need of a user.

- **Validation** Validation confirms that the needs of an external customer, stakeholder, or user of a product, service, or system are met. In software engineering, validation is the process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

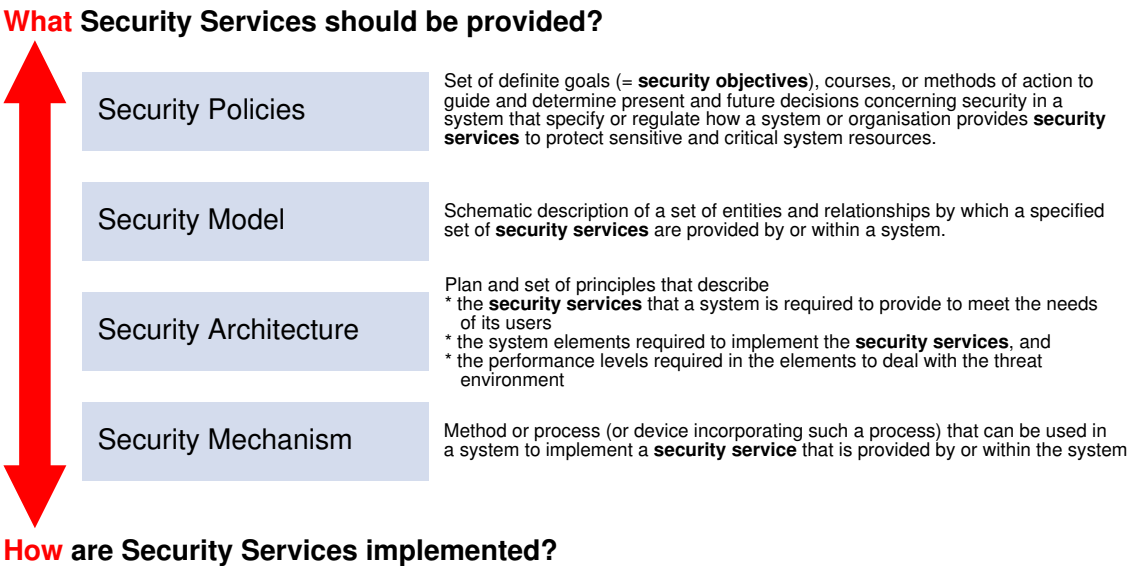

Figure 2.1: The role of *Security Services* in design, modelling and implementation for the engineering of security solutions as encompassed in RFC4949.

- **Virtual network (or cloud) provider** A virtual network provider may combine the service and resource offerings of several network/infrastructure providers (both horizontally and vertically) and offer it to a third party (the end user, or another virtual network provider). Virtual network provider and cloud provider are used interchangeably.
- **Virtual network (or cloud) user** A user that requests, uses, and consumes (shared) virtual network resources provided by a virtual (or cloud) network provider. This covers cases like Mobile Virtual Network Operators (MVNOs), or enterprises with their corporate network, or service/application/content providers (such as social media/community networks). As for provider, virtual network user and cloud user are used interchangeably.

3 Guidelines

This chapter presents general principles and guidelines that can be applied to all architecture work being done in SAIL. We also introduce frameworks for each of the Themes identified for the project: Security, Network Management, Inter-provider and Prototyping and Experimentation.

The coordination of the Themes is anchored in WP-A and the Project Management Team (PMT). Supporters have been nominated and a resource budget for the support activities is available in each WP.

In addition to the four Themes, migration guidelines are also described as migration represents an important aspect of the SAIL project.

3.1 General principles and guidelines

Rather than trying to define here guidelines that would be specific to only one of our WP, we tried to raise the abstraction level and identify the principles and guidelines that could be applicable to all these architectures. We leave to each WP the care of identifying the principles and guidelines which are specifically applicable to their problem domain.

There have been a lot of studies both from industrial and academic organisations on how to establish good network, system, and software architectures. One major publication is The Open Group Architecture Framework (TOGAF) [8], which is the evolving result of discussion between major IT players and academics from all over the world. IBM and Microsoft, which contribute to the Open Group, have also made public such guidelines [13,14]. IETF has proposed some guidelines and principles for the Internet architecture [15,16]. Some work is also currently ongoing at Europe's Information Society to identify the architecture principles of the future Internet [17].

The 4WARD project also established an abstract architectural framework for the Future Internet [18]. Many of the results from 4WARD have been reused in the specific WP architectures, mainly in WP-B.

The objective of the current section (and document) is not to repeat the numerous studies. It is rather to pick among the major architecture principles those that are the most applicable to the current project.

These principles are general rules that support the way in which the SAIL project sets its vision and values about fulfilling its mission. Architecture principles are fundamental to making decisions related to the design and construction of the SAIL architected system and can reflect a level of consensus across the project about the SAIL target system and the project approach.

We explicitly kept this list of principles and guidelines to a bare minimum to prevent putting too much constraints on the architecture work that could limit the range of potential solutions.

The principles have been split in two categories: technical and non-technical. For each principle, we also provide its motivation.

Non-technical Principles

There are many non-technical considerations on the impact of the Future Internet. They include social impacts, regulatory considerations, business models. These are studied in a separate report from SAIL [19].

The non-technical principles ended up to be a short list. We did consider to include more generic principles to take into account social aspects of the Internet such as environmental impact or digital divide. We concluded that these principles would either have little applicability given the current degree of abstraction of the technical solutions for the first class of principles or have very low acceptance among some of the project stakeholders for the second category.

Constant Change

Principle The only thing that can be taken for granted is that the Internet is changing.

Rationale Whatever we perceived as the possible foreseeable future, there will be technologies, applications, business roles or regulations that will pop-up and that will put new unexpected considerations that may deprecate principles that are deemed of most importance today.

Business Model

Principle The architecture shall support a variety of business models.

Rationale No one can predict how the technical solutions for the Future Internet described in SAIL will be deployed and who will be the main drivers for these technologies. The technical architecture should not constrain innovation and creation of new businesses or new business models and it should be able to adapt to innovations both from the technical and business sides.

Legal Constraint

Principle The architecture should not break or force applications to break legal regulation.

Rationale To get regulatory organisations to buy in the technological solution proposed by SAIL, it is of primary importance that the architecture allows to support legal constraints imposed by countries and international regulatory bodies. This includes but is not limited to privacy and intellectual property rights. In many cases, however, the legal constraints need to be resolved at the application level rather than the network level.

Technical Principles

The technical principles address mainly an openness of the solution to allow for evolution over time yet ensuring sustainability of the achieved results.

Technology Independence

Principle Architecture should not depend on specific, existing, or to-come technologies.

Rationale Technologies come and go. New technologies are introduced and old ones are made obsolete. The services defined in SAIL should operate in a large variety of technologies, current or still to be invented. Actual implementation of the architecture will have to make technical choices based on current technologies. Nonetheless, the architecture should be at an abstraction level that can be implemented with evolving technologies.

Application Independence

Principle The architecture should not favour a specific application.

Rationale Future Internet should allow all types of applications to be deployed. New applications will be created to fulfil new needs or to replace existing ones. Favouring a specific set of current applications might hinder the introduction of new ones.

Security and Privacy Built-in

Principle The Architecture should avoid retrofitting security and privacy.

Rationale It has been proved that retrofitting security and privacy will lead to sub-optimal results. If, for possible good reasons, security and privacy could not be included early on; alternatively, a validation should be carried out to assess the potential impact of not having security or privacy addressed.

Reuse

Principle When possible, try to reuse protocols and systems rather than reinventing new ones.

Rationale There are (too?) many existing standards, protocols, and systems that offer a multitude of functions and services. The reuse of existing protocols and systems eases the path to the deployment of the architecture as it allows a smooth integration with existing systems and a reduced effort of development.

Open Standards

Principle Interfaces between systems should be based on publicly available standards.

Rationale Standard protocols should be preferred to APIs to realise interfaces as they provide a looser coupling between systems and facilitate migration path.

Separation of Concerns

Principle Layering and abstraction should be used, where appropriate, to support separation of concerns between functional entities.

Rationale A clear separation of ownership and control over data and resources usually reduces the complexity and interactions between the components of a system.

SAIL being an Integrated Project (IP), it is of major importance that the different systems studied in SAIL integrate in a coherent overall architecture. This requirement has to be counterbalanced with considerations for gradual deployment. This is expressed in the following two principles.

Integration

Principle The architecture of each system defined in SAIL (NetInf, CloNe, OConS) should allow that system to integrate smoothly with the two other systems.

Rationale That is the purpose and benefit of having them studied as part of the same project.

Autonomy of Deployment

Principle The architecture of any system defined in SAIL (NetInf, CloNe, OConS) should not be dependent on the presence of the other systems.

Rationale While it is important to identify and define the integration points between the systems, to offer a smooth migration from the existing networks, each system should be able to be deployed without the presence of the other systems.

3.2 Security Framework and Guidelines

The *Security* Theme addresses security and privacy. It is organised in SAIL as a thematically oriented, coordinated activity across the technical WPs.

3.2.1 Objectives

The objectives of the Security Theme were defined as follows:

- to create a coherent and comprehensive security framework in which the project results can be evaluated regarding common security objectives and coherent security solutions,
- to understand what potential misuse could be and what remaining deficiencies are.

The minimum to achieve is firstly that the security solutions in the WPs are free of contradictions and duplication of work. Secondly, this may encompasses a gap analysis, identifying missing functions to achieve given security objectives. Additionally, as security and privacy-ensuring solutions may sometime create trade offs with usability, the theme raises awareness for these dependencies.

3.2.2 Approach

As common approach to the projects activities in cloud networking, wireless access and information centric networks, the WP individual security objectives were identified for presenting protection targets on comparable level. WP-specific security objectives are presented and discussed below (see page 32). In the following project phase the Security Theme will address how these security objectives are implemented.

3.2.3 Guidelines

The following lists some security guidelines that were identified. They are not derived from the SAIL architecture work but applicable to it. They have emerged over years and have proven themselves in best practices.

Use Security Services to Express Security Objectives

Guideline All properties concerning security objectives are expressed in terms of security services.

Rationale Security services are authentication and authorisation plus Confidentiality, Integrity and Availability (CIA). This helps expressing security objectives on comparable level.

Cryptography to Implement Security Services

Guideline Viable implementation of security services will use and only use cryptography in achieving security by design, as opposed to the approach security-by-obscurity.

Rationale Hiding secrets is short sighted and has tremendous disadvantages compared to mathematical properties that ensure that secrets can not be computed in polynomial time in cryptographic analysis.

Privacy vs. Security

Guideline Be aware that Security and Privacy may create a trade off.

Rationale Security and privacy may impose contradicting requirements. E.g. some privacy enforcing technologies may be in contradiction to accountability. The SAIL project is aware of this and attempts to design solutions that are balanced and clearly discussed and presented to allow informed decisions.

Priority for Existing Security Solution

Guideline Any solution must first use existing and well-proven standards (reuse). Only if unavoidable new solutions are invented.

Rationale Existing security standards do not only represent an often considerable amount of investment. They most often have shown and proved that they behave according to their specification and can thus be trusted to achieve the security properties that they specify. It takes additional effort and time to assess a new solution, if it has the same quality or not.

Select Security Requirements Carefully

Guideline To avoid over-dimensioning security solutions, a threat analysis and/or attacker model should be developed. This identifies suitable security services and help to design their implementation.

Rationale When gathering security requirements it is often observable that a shopping card mentality appears and identifying and expressing requirements is somewhat excessive, because resulting consequences are not clear or experienced. In the end one has to pay the filled shopping card. Methodologies employing e.g. a threat analysis have helped a lot to right-size requirements.

Don't Hinder Security Management

Guideline Be prepared for and do not hinder security management to enable sustainability of the solution.

Rationale Security Management maintains the protection level and helps to ensure that security properties do not fade out over time, e.g. because of an increasing remaining risks or new/enlarged vulnerabilities.

3.3 Network Management Framework and Guidelines

Management in general is concerned with questions about how to manage each of the heterogeneous network and IT resources in the SAIL architecture both individually and in conjunction with other resources. While individual work packages each define specific management concepts, the network management theme focuses on the latter, providing the conceptual "glue" that allows to consistently combine resources in terms of management across SAIL's overall architecture.

3.3.1 Objectives

According to the SAIL proposal's Annex I - "Description of Work", the main objective of the network management theme is to **"define [...] an overall management architecture based on [the] decentralised self-management paradigm".** For that purpose, the network management theme proposes a set of guidelines that enclose general management procedures and interfaces that should be implemented by the individual architectures of WP-B, WP-C, and WP-D, so that management functions can be implemented consistently across these architectures. In order to not hinder the implementation of management functions that may be specific to some network domain and that do not need to be shared across SAIL's overall architecture, the following proposed guidelines are only recommendations. However, only those management functions that adhere to any or all of the guidelines are able to integrate with the overall management architecture.

3.3.2 Interrelations

The definition of an overall management framework is dictated by some basic interrelations on project, work package, task, and theme level in the first place. Figure 3.1 illustrates the relations that the network management theme's architecture takes into account. In the figure, key example architectural elements are indicated for each work package and some of the themes (the security theme does not have a specific component associated with it). Note that only conceptual themes are shown, which excludes the prototyping theme.

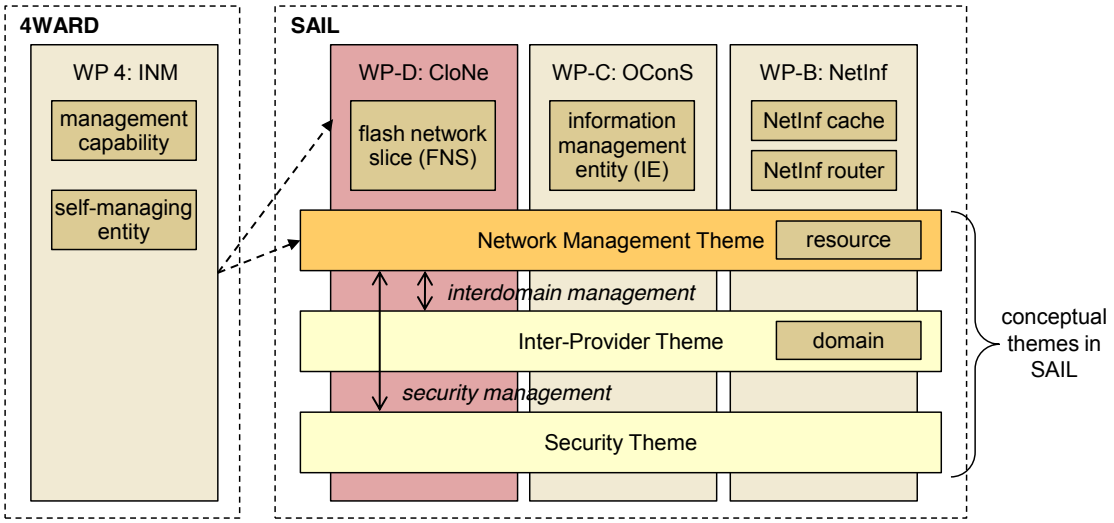

Figure 3.1: Interrelations on the project, work package, task, and theme level.

Firstly, the network management theme is centred around the concepts developed in WP-D's management task. This task is concerned with a broad functional scope including goal translation, fault management, and resource management, hence defining a suitable baseline for the establishment of guidelines that apply to the wider architectural scope across all of SAIL. We will show that some of the concepts of WP-D are at the core of a number of the following guidelines, since they are naturally compatible with the architectural concepts of other WP's.

Second, Figure 3.1 indicates the primary resources that are dealt with in other WPs and themes. In WP-B, the main manageable resources comprise the NetInf Cache (NC), the NetInf Router (NR), and the NRS [2]. In WP-C, the main resources of consideration are the Information Management Entity (IE), Decision Making Entity (DE), and Execution and Enforcement Entity (EE) [3], which are embedded within network elements, hence designate the manageable entities.

Third, we consider the conceptual themes, which have a close relation to the network management theme. The inter-provider theme focuses on the domain concept, which is an essential abstraction also in each of the technical work packages. Hence, inter-domain management questions are to be aligned between this theme and the network management theme. The security theme is primarily concerned with technical approaches to information privacy and ICT system protection. Since security management is a natural part of any management architecture, coordination is also required between this theme and the network management theme.

Fourth, management in SAIL is inspired by the concept of self-management, which is defined in the 4WARD project under the term of In-Network Management (INM) and which applies autonomous distributed management principles to individual network elements via the principal management elements termed Management Capability (MC) and Self-Managing Entity (SE) [20], [21], [22]. In SAIL, some of these concepts are applied to different architectural elements: in WP-D,

these elements are flash network slices (FNS) [4]; in the network management theme, these elements comprise resources in general (see the second guideline for how to interpret resources).

3.3.3 Guidelines

In the following we propose a set of architectural guidelines for management in SAIL that is founded on results of WP-D's management task [4] and an analysis of WP-B's and WP-C's architectures as defined in the current work package deliverables in [2] and [3], respectively. Because the architectures in each of the technical work packages will be refined further, the guidelines may be modified and amended accordingly as the SAIL project progresses. We note that the first guideline is of nontechnical, whereas the following guidelines are of technical character.

Any of the following (technical) guidelines is non mandatory

Guideline Any of the following technical guidelines is in principle optional and components of the overall architecture do not have to implement them.

Rationale This nontechnical guideline states that any of the network elements and other components present in a productive networked system according to SAIL's architectural framework does not necessarily have to adhere to any or all of the guidelines. This guideline is straightforward yet key in providing a flexible management system that can evolve and migrate over time and that does not force all stakeholders to adopt a rigid management approach from the very beginning. For example, upon the introduction of new networking technology (a relevant example in the context of SAIL is OpenFlow switching technology) that is still under development and its reliability may still be assessed, a stakeholder may not wish to readily introduce the technology's resources into the overall productive resource pool for creating complex flash network slices. Further, whether a stakeholder wishes to implement any or all of the guidelines may also depend on mutual benefits it can achieve with other stakeholders. This is a well-known concept which is also found, for example, in peer-to-peer-based information sharing, where a user is able to download only when uploading at the same time. Sharing and using resources between stakeholders will likely be subject to very similar laws, and it can be expected that adherence to guidelines and consequently, the extent by which management knowledge and functions are shared by individual stakeholders, will self-organize.

All architectural elements deemed manageable must be modelled as manageable resources

Guideline Any element that is part of the overall SAIL architecture and that requires management consistent with these guidelines must be modelled as a manageable resource.

Rationale In order to provide the homogeneous use of management functions, at the very basic level, each architectural element that is to be managed should be modelled as a manageable resource. This guideline is inspired by the architecture in WP-D, which defines networking, processing (also termed computing), and storage as the basic resources for the composition of flash network slices (FNS). The idea is that a resource possesses a number of well-defined properties that enable it to be used consistently within the management framework. Two important such properties are a resource's unique identification and its resource description. For example, resource discovery, a fundamental management function defined in WP-D, requires unique resource identification for resource enumeration, and fault management (also defined in WP-D) requires homogeneous resource descriptions to be able to negotiate with resource management on the replacement of a faulty resource that is, for example, part of a flash network slice. In Figure 3.1, for WP-C, the

information management entity (IE) and for WP-D, the NetInf cache and NetInf router, must be modelled as a resource if this guideline is to be satisfied.

The context of a resource is to be published in order to allow extended management

Guideline The context of a resource is to be published and described alongside with the manageable resource in order for extended management functions to become possible.

Rationale While the previous guideline provides information about the resources a stakeholder is able or willing to share in terms of management, this guideline requires that additional properties of the resources' context are to be published. Usually, this context corresponds to the domain within which the resource is located, such as the NetInf router of a network provider. The context may also correspond to other suitable forms of abstractions, such as network layers (e.g. the NetInf layer, see Figure 3.3 in [2] and Figure 5.7), which is a further suitable form of structuring e.g. within a domain. The context may define any conditions or restrictions that are applicable when manageable resources within the domain are used. The context covers domain-specific knowledge that is relevant for handling that resource in terms of management. A typical example that also relates to security is the fact that encryption may be subject to legal restrictions, more concretely, it may allow only certain upper bounds on the bit length of encryption keys. If two domains use different such restrictions, a common denominator may later be chosen that consistently combines the resources from different domains such that legal restrictions are satisfied.

So far, guidelines address manageable resources and their context. This allows the usage of resources in their correct context for management purposes, but it does not allow the use of management functions that may have been defined in some part of the overall SAIL architecture. The following guideline concretises the functional part.

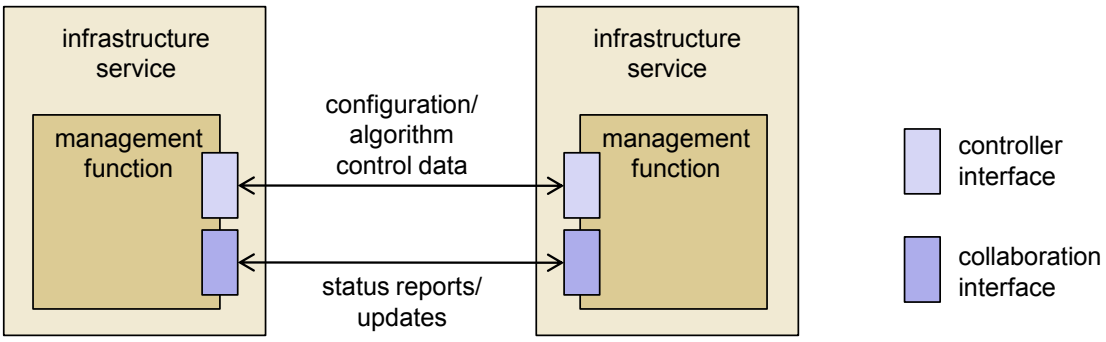

Figure 3.2: Interaction between management functions via WP-D's two types of interfaces.

Management functions shall be implemented and published by making use of the two types of management interfaces defined in WP-D, namely, the controller interface and the collaboration interface.

Guideline Each management function that is provided by any of the architectural components shall make use of the controller and collaboration interfaces defined in WP-D, which define how management functions communicate with one another.

Rationale WP-D defines in [4] a specific approach of how management functions are to communicate with one another in order to build more complex management functions from composition. In Figure 5.2 in [4], which we show for reference in a simplified form in Figure 3.2, two generic interfaces are defined as follows. On one side, according to [4], via the controller interface, management services and algorithms are created, configured and destroyed.

On the other side, via the collaboration interface, continuous information to relevant recipients, such as updates about resource states, reports of potential faults etc. are published.

Typical examples for management functions in WP-B are distributed cache management, which is distinguished into caching at the network edge and in-network caching (Section 2.1.5 in [2]), and management of persistent data storage (Section 2.1.6 in [2]), which closely relates to storage resources that are also a basic manageable resource in WP-D. In WP-C, one example of an important management function is connection management, where the connection between different endpoints is appropriately managed throughout the lifetime of that connection.

A common data model shall be used to describe manageable resources, contexts, and functions.

Guideline A common data model, sufficient to express all relevant aspects of an overarching management architecture, shall be used to describe all manageable resources, their contexts (e.g. domains, layers), and management functions.

Rationale A data model that is understood by an overarching management architecture is essential to be able to build complex management solutions by means of composition. Typically, such a data model should be standardised, in order to achieve basic agreement between different stakeholders. Especially in the context of SAIL, stakeholders are diverse, ranging from hardware manufacturers to solution providers, each being involved in some aspects of management, especially resource management. One part of this data model shall be based on WP-D's data model, described in Chapter 3 in [4]. Furthermore, such a common data model is essential in order to build the knowledge plane illustrated in Figure 5.3 in [4].

3.4 Inter-provider Framework and Guidelines

This section describes the work that needs to be done in the work packages of the SAIL project in order to come up with clean and future-proof definitions and implementations of the domain concept and inter-domain interfaces. The guidelines developed in the Inter-provider Theme will guide the relations between providers. They also apply within a provider, when different network domains are identified and need to be cleanly isolated.

3.4.1 Objectives

The main objective of the Inter-Provider theme is to make sure that the solutions proposed by SAIL overcome the lack of attention given in IP with regard to network domains: the Internet was born as a flat network and provider domains were included as a *patch*. The protocol governing inter-provider routing information exchange in the Internet is BGP-4 [23]. Despite being *the* Interdomain routing protocol of choice, it has flaws that have affected the integrity of the whole Internet with routing storms, traffic diversions through malicious or just careless operation, etc.

The *Interprovider* Theme will make sure a common understanding of the domain concept is achieved in SAIL. This common understanding will be the basis for the definition of the domain concept in the different work packages. Once this objective is achieved, the Theme will derive the functionalities needed for the inter-domain interfaces.

3.4.2 Guidelines

The Interprovider Theme is unifying the work on the interfaces between domains (as understood by each WP). When looking at the current Internet's issues, the three topics that need an urgent

answer are:

1. Definition of the domain concept
2. Definition of the information exchanged between domains
3. Trustworthiness of the information published by a domain

These issues have resulted in the following guidelines:

Each technical WP must have a definition for a domain

Guideline Each WP is working at different levels in the communication stack and the concept of domain may vary depending on the context. However, there must be a clear definition of the domain concept in each work package.

Rationale Inter-provider work needs to be embedded in the work packages. Thus, each will need to establish a common understanding of what domain is. This work is progressing correctly. Chapter 2 includes the terminology used in the different WPs.

Information exchanged at the border between domains

Guideline Each work-package must identify a minimum set of information that needs to be published in the interdomain interface in order to make inter-working between domains possible.

Rationale An example of current problems in the Internet regarding the information advertised by Autonomous Systems (ASes) is fragmentation: ASes do not advertise the best aggregations of their assigned prefixes and bloat up the routing tables in the Internet's Default Free Zone (DFZ) [24].

Information published at inter-provider interfaces needs to be trust-worthy

Guideline The WPs need to make sure that the information provided by one domain to other domains is legitimate and will not induce traffic diversions or have any other adverse effects on the overall network.

Rationale One of the main problems of today's BGP-4 implementation is that despite proposals to introduce packet signatures, Public Key Infrastructure (PKI) infrastructures, etc., it is still impossible to make sure that the prefixes advertised by an AS are not assigned to another AS. This practise has recently been denounced before the US Congress.
This topic will need to be addressed in cooperation with the Security Theme.

3.5 Prototyping and Experimentation Framework and Guidelines

3.5.1 Approach

In this section, we discuss principles, guidelines and requirements for the SAIL prototyping and experimentation (validation) activities, as they are managed by the cross-functional Theme *Prototyping and Experimentation*. The objective in this theme is to ensure the necessary project-wide coordination and consolidation of prototyping and experimentation activities in order to foster a roadmap for the practical activities across the work packages. This covers several stages from early partner-specific experiments, to clustered WP prototypes, finally resulting in a project-wide demonstration at an industry-relevant event by the end of the project.

The SAIL project aims at compelling, prototype-based evidence that its research results and innovations will be a core part of the Network of the Future. To achieve this by the end of the project, we use a more agile approach than usually done in a research project. In a first phase (until month 18), the prototyping activities have early started from a bottom-up perspective, both

at partner and WP levels, with minimal overall planning and cross-task specification work based on this coordinating Theme. This allows an immediate transfer of experience gained in early phases to the second project phase (month 19-30). This phase is then characterised by an increasing effort on inter-task and cross-WP cooperation in the form of cooperative federation rather than full integration of the developed system components. By this approach, we minimise the overall dependencies and risks and allow the project to adapt to theoretical achievement and progress concurrently and quickly – leaving the freedom to change and adapt to creative input.

3.5.2 Objectives

The goal of the experimental approach in SAIL is to prove that the SAIL research results are mature innovations satisfying the initially postulated expectations. This includes:

- either, demonstrate innovative (i.e differentiating) solutions (functions, features, scenarios) to known/existing requirements or problems
- or, demonstrate basic (unique, first) solutions (functions, features, scenarios) to innovative requirements or problems.

Credible evidence for these innovations shall be achieved by practical validation in multiple stages:

- By **prototyping** that can be used to validate specific aspects (properties) of innovative model, system or component designs against the initial assumptions (requirements). A prototype typically simulates only a few aspects of the features of the eventual technical system. Basic prototype categories are, among others, proof-of-concept prototypes (checking feasibility), functional prototypes (checking required functions), or visual prototypes (checking look-and-feel).
- By **experimentation** as a means in the scientific methodology that arbitrates between competing models or hypotheses. Experimentation is also used to test existing theories or new hypotheses in order to support or disprove them. However, an experiment may also test a question or previous results. From that, it is clear that experimentation requires a realisation of a prototype of any kind.
- By **demonstration**, which is a scientific/technical experiment carried out for the purposes of proving scientific/technical effects or solutions, rather than for hypothesis testing or knowledge gathering (although it may originally have been carried out for these purposes).

SAIL does not aim at conducting field trials or massive test deployments in operating network environments.

3.5.3 Principles

The stages as described above require not only a prototype realisation of the innovative solution, but also a validation framework that is used to control the test environment and allow identifying initial - as reproducible as possible - conditions, including:

- Setting the environment conditions and control the pre-conditions;
- Variation of input / modification of [scenario, network, component] configuration;
- Measurement and visualisation of output / [network, component] behaviour.

Hence, the following **principles** are identified as essential for the validation framework:

Control of Pre-Conditions

Principle The architectural framework for validation shall be able to control and identify relevant environmental conditions and system pre-conditions.

Rationale This is necessary in order to allow a reproducible or comparable course of experiments or demonstrations. Some degree of automation would be preferred for the purpose of regression tests.

Observe System Behaviour and Output

Principle The architectural framework for validation shall be able to measure, monitor and visualise relevant system reactions and outputs.

Rationale This is necessary in order to enable comparison of expected and actual system behaviour, as well as a root cause analysis in case of failure. Some degree of automation would be preferred for the purpose of regression tests.

3.5.4 Architectural Framework for Validation

Therefore we sketch an architectural framework (see Figure 3.3) that can be used as a model across all work packages to coordinate their experimental activities, and thus can drive further cooperation initiatives. Besides the main components, i.e. the *Component under Test/Consideration* and its control and management component, we identify the need for (i) a *Test Management* component to initiate and configure the test system, (ii) a test application component driving application scenario sequences and (iii) and a evaluation component to monitor, display and visualise the system behaviour. The use of specific tools for the comparison between expected and actual behaviour will depend on the specific scenario, since it might suffice with pure appearance.

Whereas the control and management interfaces as well as the data/service interfaces of the component under consideration are the interfaces described upon the guidelines of this architecture document, the interfaces T1 to T4 are interfaces specifically needed for validation purposes and cooperation between common parts of the environment.

Typically, the validation environment will require realisations in a distributed network environment involving multiple nodes and links, however virtualisation techniques could be used to integrate the different parts on a common or unified platform.

Within this framework, the following **technical challenges** need to be considered in more detail:

- How to extend a cooperation and interoperability concept of components to a interoperability concept of the validation environment?
- How to go from a (partner) component demonstration to a WP demo?
- How to go from WP demo to project-wide demo?
- How to interface/combine/integrate cross-WP test applications and run-time environment?
- What are the candidates for use in a common validation environment, e.g. re-combination of test applications, traffic generation tools, monitoring tools, visualisation tools?

3.5.5 Guidelines on Validation Process and Framework

One of the project goals is a project-wide demonstration at an industry-relevant event by the end of the project. Even though this is not necessarily based on an integrated prototype, it will require a project-wide coordination of the scenarios and use-cases that are subject to respective prototypes

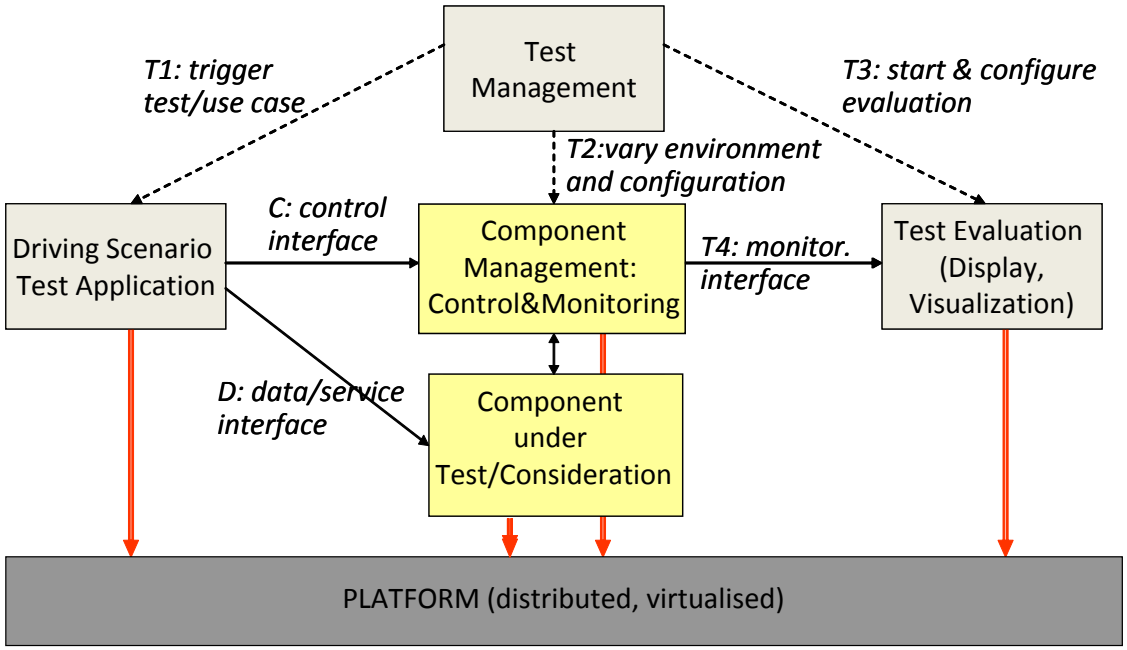

Figure 3.3: Prototyping, Test and Experimentation Framework

and their realised features. A detailed description and specification of the related inter-WP topics will be given in the Deliverable DA.9 due in month 21.

Coordinated Realisation of Inter-WP Scenarios and Use Cases

Guideline Features of prototypes realised at a WP level shall follow the common storyboard outlined in the Deliverable *Description of project wide scenarios and use cases* [25] in order to assure functional compatibility across the WPs (at least pairwise over two, however preferably over all WPs).

Rationale Without coordination of the features to be realised, there is the risk of disjoint commonalities between the WP parts.

In order to ease cooperations between the project prototype parts, the following technical guidelines on the prototyping environment have been identified:

Realisation of Inter-WP Interfaces

both for the executing components as well as for the test environment components

- The inter-WP interfaces shall be lightweight, easy to describe and to implement.
- The inter-WP interfaces shall be extensible on a peer-to-peer basis, avoiding the effort for overall project-wide specifications.
- The technology for realising inter-WP interfaces shall be supported by many commonly-used development environments, not posing strong restrictions on availability and purchasing specific tools.

Use of Virtualisation Techniques

- Virtualisation techniques shall be considered as a means to abstract from heterogeneous hardware and software platforms and increase the opportunity to cooperate between project

parts.

- The selection of virtualisation tools shall take into account their availability on many commonly-used OS/HW platforms.
- Virtualisation techniques shall be used to up-scaling of trials with limited HW resources.
- Virtualisation techniques shall be able to integrate physical environments and devices as necessary (e.g. use of physical mobile devices in an heterogeneous wireless access environment)

Provisioning of Common Testbed/Network Facilities

Common testbed/network facilities for the project-wide experiments and demonstrations would be desirable, under the following requirements:

- Common testbed/network facilities shall provide open access for project partners from remote.
- Common testbed/network facilities shall take into account the need for (trans-)portability for demonstration purposes and the access availability at these locations .
- Common testbed/network facilities shall provide virtualisable network resources, possibly also beyond (below) IP layer functionality.

The following topics describe the procedure to be followed for a best practice experimental design and are given here as **guidelines on the validation process**:

- Problem statement: which aspect is under consideration? problem to be solved, hypothesis to be tested, aspect/property to be validated
- Describe model/realisation assumptions and abstractions, constraints
- Identify system pre-conditions and input parameters
- Specify expected experimental outcome
- Run prototype test, experiment or demonstration
- Document actual experimental outcome
- Compare expected outcome vs. actual outcome, and evaluation: passed or failed

3.5.6 Guidelines on Free and Open Source Software

3.5.6.1 Approach

The use and/or distribution of Free and Open Source Software (FOSS) (as opposed to proprietary code) has numerous advantages to the project partners, including cost savings (by elimination of royalty and deployment license management), increased productivity and flexibility, increased software reliability and stability (due to more users/developers/testers in the open source community providing feedback, bug fixes and testing), increased visibility and accelerated dissemination, among others. This applies in particular for the use of FOSS in the SAIL project prototyping, experimentation and demonstration activities.

There are also disadvantages, including the complexity of FOSS licensing (e.g. when a FOSS distribution has multiple dependencies, i.e., consists of multiple software packages, each subject to different license terms), increased risks or liabilities due to the absence of warranties and indemnities normally provided for proprietary software, deleterious impacts on certain commercial partner

interests, and potential issues associated with ownership of intellectual property originating from unknown and questionable sources, etc.

The purpose of these guidelines is to maximise the benefits of FOSS to the SAIL project while mitigating the associated risks.

In this context, FOSS is defined as:

- software provided to Licensor royalty-free in source code form, under a license including, but not limited to, one approved by the Open Source Initiative (OSI)[1] or
- proprietary software provided to Licensor royalty-free in binary code form, under an end user license agreement that is accepted without a signature, or
- shareware provided to Licensor free of initial charge, such as on a trial basis, but where a fee may become due once the user decides to use the software beyond the trial period, or
- public domain software

There is no single, universally used definition for FOSS. Here the term FOSS comprises both software provided free of charge as well as software provided in open source or closed source fashion, and any combination thereof. It attempts to capture that software which, from a corporate perspective, presents particularly noteworthy risks because it can be obtained, in most cases, without signing a written license agreement, and the license agreements applicable to it tend not to be thoroughly reviewed in all cases.

3.5.6.2 Legal Guidelines

The project partners shall be committed to carefully managing the use of Free and Open Source Software (FOSS), both within the project consortium and when releasing and publishing code as a project result.

This includes as a legal obligation of each partner:

- Identify any FOSS components intended for use within the project consortium and when releasing and publishing code or in proposed service or product offerings as a project result.
- Ensure that FOSS license obligations are met and do not interfere with the Intellectual Property Rights (IPR) and Access Rights of the consortium agreement.
- Manage FOSS according to the project publication rules of the consortium agreement, when publishing, subcontracting, outsourcing, modifying and/or contributing FOSS to open source projects.
- In project-internal partner cooperations, ensure that use of FOSS by one partner does not impact IPR or copyright of other project partners or has undesired/unexpected impacts (e.g. need for disclosure or publication of project results) on these.

A list of most common license terms and conditions, ordered by name can be obtained from the OSI Website [26].

[1] `http://www.opensource.org/`

3.5.6.3 SW architectures involving GPLed FOSS parts

We discuss briefly the risks when using FOSS under GNU General Public License (GPL) [27] and the implication on SW architectures involving GPLed parts, as it is a commonly used model (e.g. Linux kernel).

The GPL was designed to be the antithesis of the standard proprietary license. To this end, any modifications that are made to a GPL program are required to be given back to the GPL community (by requiring that the source of the program be available to the user) and any program that used or linked to GPL code is required to be under the GPL. The GPL is intended to keep software from becoming proprietary. As the last paragraph of the GPL states: "This General Public License does not permit incorporating your program into proprietary programs" [28].

From this, we conclude that using GPLed FOSS within the project may not be compliant with the SAIL consortium agreement, because of the risk of so-called GPL contamination, i.e. its impact on other partner's software licensing rights, forcing it to be also under GPL.

The GPL is a complex license, so here are some rules of thumb when using GPLed SW with implications on software architecture (but without any legal warranty and without claiming completeness):

- The Licensor can charge as much as he want for distributing, supporting, or documenting the software, but he cannot sell the software itself.
- The rule-of-thumb states that if GPL source is required for a program to compile, the program must be under the GPL. Linking statically to a GPL library requires a program to be under the GPL.
- The GPL requires that any patents associated with GPLed software must be licensed for everyone's free use.
- Simply aggregating software together, as when multiple programs are put on one disk, does not count as including GPLed programs in non-GPLed programs.
- Output of a program does not count as a derivative work. E.g., this enables the gcc compiler to be used in commercial environments without legal problems.
- Since the Linux kernel is under the GPL, any code statically linked with the Linux kernel must be GPLed. This requirement can be circumvented by dynamically linking loadable kernel modules. This permits companies to distribute binary drivers, but often has the disadvantage that they will only work for particular versions of the Linux kernel.
- The common interpretation is: code that runs in the same address space as GPL code, is contaminated, which in turn means:
  - Linux application running in user address space: no contamination
  - Linux drivers running in the kernel address space: contamination
  - Static linking: GPL libraries are contaminating
  - Dynamic linking: GPL libraries may be contaminating. However, dynamic linking is still not very clear in GPL-V2, but has been clarified in GPL-V3 in that the requirements that must be fulfilled for a contamination to occur are more severe than in GPL-V2.
- Communication via remote procedure call, socket, pipe, command line arguments: no contamination
- In object-oriented languages: Sub-classing GPL classes is derivative work: contamination

In more complex cases, legal expert advice and evaluation is necessary, which cannot be given here. Some hints may also be found in FAQ lists [29].

From the above examples, it should be clear that carefully identifying and managing licenses of FOSS to be used in the project prototyping, experimentation and demonstration software as well as a conscious SW architecture design avoiding possible license clashes is in the interest of each partner in order not to infringe contractual obligations.

3.5.6.4 FOSS licenses that are less stringent than GPL variants, e.g. NewBSD

BSD licenses are a family of permissive free software licenses. The original license was used for the Berkeley Software Distribution (BSD), a Unix-like operating system after which it is named. The first version of the license was revised, and the resulting licenses are more properly called modified BSD licenses. Two variants of the license, the New BSD License/Modified BSD License [30] and the Simplified BSD License/FreeBSD License [31] have been verified as GPL-compatible free software licenses by the Free Software Foundation (FSF)[2] and have been vetted as open source licenses by the Open Source Initiative, while the original, 4-clause license has not been accepted as an open source license and, although the original is considered to be a free software license by the FSF, the FSF does not consider it to be compatible with the GPL due to the advertising clause [32].

The BSD licenses have fewer restrictions on distribution compared to other free software licenses such as the GPL or even the default restrictions provided by copyright, putting works licensed under them relatively closer to the public domain.

The BSD License allows proprietary use, and for the software released under the license to be incorporated into proprietary products. The New BSD License/Modified BSD version allows unlimited redistribution for any purpose as long as its copyright notices and the license's disclaimers of warranty are maintained. The license also contains a clause restricting use of the names of contributors for endorsement of a derived work without specific permission.

Works based on the material may be released under a proprietary license or as closed source software. It is also possible for something to be distributed with the BSD License and some other license to apply as well.

This is the reason for widespread use of the BSD code in proprietary commercial products, and may therefore be the right choice for licensing code generated under the SAIL consortium agreement.

For an **example** how to deal with (acceptable) licensing in the context of a FP7 project, see the pages of OpenNetInf related to some code released as a result of the 4WARD Project[3]:

- licenses of used software:
  `http://www.netinf.org/opennetinf/libraries-licenses/`
- licenses of code generated and published in relation with the 4WARD Project:
  `http://www.netinf.org/opennetinf/source/` and
  `http://code.google.com/p/opennetinf/`

3.5.6.5 Technical Guideline

As a conclusion from the discussion above, the following **technical guideline** should be noted:

Software Architectures Involving FOSS

Guideline The software architecture for the SAIL prototyping, experimentation and demonstration activities shall be aware of the FOSS licenses involved in the realisation, and shall design

[2] `http://www.fsf.org/`
[3] `http://www.4ward-project.eu/`

interfaces and mechanisms in order to separate software components in a way as to avoid proliferation and contamination of license conditions across the boundaries, especially between partner contributions.

Rationale Conclusion from the previous discussion.

3.6 Migration and Interoperability Framework and Guidelines

This section is intended to identify the most relevant migration topics and issues, with a view to define a migration path for the technologies being developed by SAIL. To start with, two main areas are proposed, as briefly described below.

3.6.1 Technical challenges

Migration often represents a major challenge and represents an obstacle against widespread deployment of new network technologies. Usually, a number of migration techniques are available:

- Overlay techniques, to enable new technologies to be deployed on top of existing infrastructure, including the Internet, while also providing distinctly novel features. NetInf applied on top of the current Internet can be taken as an example of such techniques; many other examples exist (e.g., P2P, VoIP services via Skype).
- Translation techniques, to allow communication between different systems or networks, usually though protocol mapping. This has been used for quite some time in the telecommunications world in the past (e.g. interworking between traditional voice services and VoIP).
- Dual-stack techniques, to allow different protocols to co-exist in the same devices and networks. IPv4/IPv6 is the obvious example here.

Deployment of new network technologies on a large scale is often hindered by technical obstacles, which may not be apparent in a small testbed environment, but become clear when it comes to deployment on a large scale. Examples of such issues with new technologies are:

- Incremental deployment: can it easily coexist and inter-operate with legacy technologies?
- Multi-domain deployment: can multi-domain scenarios be supported, including interoperability with/across legacy domains?
- Standardisation gaps: are there any gaps in standardisation that may preclude its widespread deployment?

3.6.2 Business incentives and obstacles

The costs for enabling migration must be carefully considered. In general, it is very difficult for a new technology to be deployed on a significant scale without sufficiently attractive business incentives for all players involved. IPv6 perfectly illustrates the difficulty to introduce fundamental changes in a large scale infrastructure (in this case, Internet) on a global scale – in spite of the long-recognised issues with the IPv4 address shortage, the deployment of IPv6 has been weak in several world regions, including Europe and North America. However, counter-examples can also be found – for example, in the early 2000s, the deployment of MPLS technology required a major infrastructure upgrade from network service providers. Nevertheless, the technology was widely adopted. Two major differences between IPv6 and MPLS should be noted. Firstly, MPLS provided a clear business case: VPNs; by contrast, IPv6 offered a number of new features, but none

of them provided a really attractive business case to be exploited. Secondly, the benefits of MPLS could be enjoyed independently by each network operator; in contrast, the benefits of adopting IPv6 will not be real until the technology has been deployed on a significant scale by other network operators. As the example of IPv6 has shown, a limiting factor against the success of new network technologies is the fact that each individual service provider will not be able to benefit from the technology migration until others do it as well, which tends to postpone the migration process and ultimately discourages the adoption of the technology. The issue of business incentives may be particularly relevant when not a mere upgrade between technologies is at stake but the potential emergence of new business models and new business players. An example where this might be the case is cloud networking, with the blurring of the traditionally clear boundary between IT service providers and network service providers and the potential emergence of new business players (e.g. service brokers, combined cloud/network service providers). In summary, the topic of migration must be analysed by WPs B–D, both from a purely technical perspective and from a business oriented perspective.

3.6.3 Guidelines

From the very beginning, the incremental deployment of results was clearly defined as an ambition of the SAIL project. In this regard, the definition of a viable migration strategy is an essential requirement of the SAIL project.

Identify Migration Roadblock

Guideline Each WP must analyse migration issues and identify potential roadblocks against deployment on a significant scale.

Rationale Past successes and failures have shown that migration issues can represent a major roadblock against the widespread deployment of a new technology.

Progressive Deployment

Guideline There should be some benefit from deploying SAIL technology no matter how small is the scale of deployment.

Rationale SAIL should avoid the pitfalls of global harmonisation requirements in terms of regulation and standardisation. Several examples can be used to demonstrate that the need for global compromises is often an obstacle against successful technology deployment.

Standardisation

Guideline SAIL should be in a position to actively contribute, and even to steer, the work on standardisation in relevant bodies.

Rationale SAIL is active in technical areas that still lack a stable standardisation framework. Any technical solutions should be developed taking into account ongoing developments in relevant standardisation bodies.

Business Model

Guideline Clear business models must be defined by all technical WPs.

Rationale There should be a clear benefit to all potential players involved (e.g. end users, content providers, infrastructure providers, service brokers).

4 Topics to be addressed

This chapter enumerates the list of topics or concerns that should be addressed by each WP architecture. It is intended to provide an informal check list for the WP architecture documents.

Offered services
Which services does the described architecture offer?

Used services
Which services does it use, and from which other systems or layers?

Applications
What are the potential applications to benefit from this system?

Main functions
What are the main function blocks of the architecture?
How are they related?

Interfaces and protocols
What are the main interfaces identified in the architecture?
Which protocols are required?

Security
What are the main security objectives?
How are they addressed?
What are the remaining risks not covered?

Network Management
What are the main challenges related to network management?
How are these challenges addressed?
How can a uniform data model for a common management architecture be achieved?
What management mechanisms can be provided to motivate stakeholders to share resources?

Inter-Provider
Which functions can span over multiple domains?
How are the responsibilities for the overall correctness of the network shared between interconnected domains?
How is a decision that affects or may affect several domains communicated?

Prototype and Experimentation
Are the functional blocks and interfaces realistically implementable?

Scenario and Use Cases
Which of the identified (WP and project-wide) scenarios and use cases were used to derive the architecture?
How do the use cases map to the functional architecture?

Migration
Has potential migration path from the existing network have been identified?

Deployment

What are the potential mapping/distribution to network nodes?

SAIL Overall architecture

How does the work package architecture fit in the overall SAIL architecture?
Which are the services used by the described work package that are offered by the work package providing the underlying layer?

Evaluation and Open Issues

Does the architecture described fulfils the objectives set initially?
What are the strengths and weaknesses of the selected solution?
What elements require further investigation?
What are the known open issues?

The following topics don't need to be addressed directly in the architecture documents as there will be specific deliverables dedicated to them.

Migration

This will be studied in *D.A.4 - Final migration description.*

Business Models

This is covered by two deliverables: *D.A.7 - New business models and business dynamics of the future networks* and *D.A.8 Evaluation of business models.*

5 SAIL Architecture

5.1 Simplified Architecture

5.1.1 Overview

Figure 5.1 illustrates a simplified view of the SAIL architecture, presenting the major relations between the three main systems developed in SAIL. One could easily imagine that more interactions can exist between each system (e.g. Cloud Networking (CloNe) using NetInf for exchanging information or Open Connectivity Services (OConS) using CloNe for allocating computing and storage resources for its control and management planes). These additional interactions have not been considered in this simplified version of the architecture because they don't represent typical use and add complexity without adding clarity. Even if the diagram illustrates a clean layered approach, the readers should not assume a strict layering when SAIL will be fully deployed.

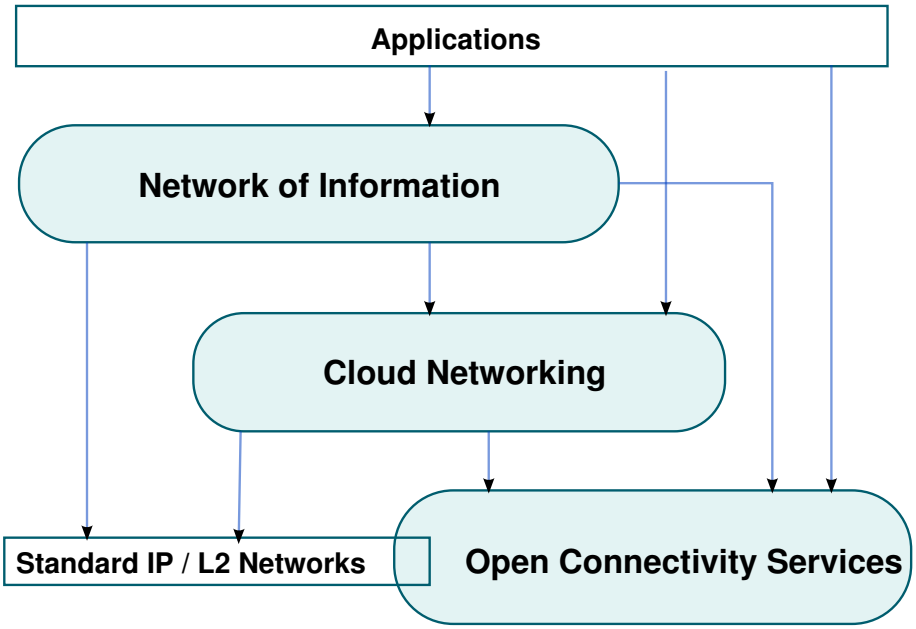

Figure 5.1: Simplified SAIL Architecture

Let us describe the architecture top-down.

- **Applications** Multiple applications types take advantage of the SAIL architecture – examples are elastic content delivery, social networking, or next generation mobile communications. Some typical use cases and scenarios are described in [25].

- **NetInf** defines an information-centric network with Information Object (IO) as the central concept rather than focusing on node communication. NetInf provides a set of services to identify, retrieve, and transport IO. The NetInf naming scheme allows to uniquely identify IOs independently of their location. Since many copies of the same IO can exist in the network, NetInf provides a distributed name resolution service to determine the location of a suitable copy of the IO and selects the appropriate transport mechanism to deliver the object to the requester. The NetInf architecture is presented in [2].

- **CloNe** manages and controls computing, storage and connectivity resources. The multi-provider approach of CloNe allows data centres and network provider to cooperate to offer a service end-to-end. Applications can benefit of placing their data and running software at the most appropriate place in the network. More details on the CloNe architecture can be found in [4].

- In a fully deployed SAIL, **OConS** owns and controls the connectivity resources. OConS offers services to extend IP and L2 networks. It provides mechanisms to deliver content the best way, using a large set of connectivity technologies. OConS Multi-P, which stands for multi-point, multi-path, multi-protocol, enables to deliver data using the best connection available, taking advantage of the fact that many devices use multiple connectivity technologies, with or without wires. OConS offers services that integrates the different layers and domains to get a more efficient data plane. The OConS architecture and services are defined in [3].

5.1.2 Interfaces

NetInf - CloNe interface

NetInf can use CloNe to allocate computing, storage, and connectivity in different parts of the network. These resources will support NetInf infrastructure (e.g. IO caching and control functions). NetInf can use the virtual infrastructure elasticity provided by CloNe to react to changing traffic and load patterns, e.g., to dynamically create a cache in the network when serving data from such a cache reduces traffic load, or deleting that cache when the demand is not there anymore.

From the perspective of CloNe, NetInf is simply another virtual network user. In that sense, it will use the interface provided by CloNe and presented in section 2.3 of [4].

One of the issues in the interactions between CloNe and NetInf (or any other CloNe user) is how to take the best decision on which resources in the network to use to fulfil the specific needs of the cloud user.

For example, in the elastic video distribution scenario presented in [25], the decision on where to put the video nodes in the network depends on the actual video traffic and end-user usage patterns (application specific) and the actual topology of the network including computing and storage devices location (network provider specific). There are two trivial solutions to this problem. The first requires that the application gets the topology information from the network provider to take the placement decision. The second solution push the decision on the other side of the interface: the application provides traffic and usage pattern to the network provider and the latter decide on the resource placement. However, none of these solutions are ideal because neither the application provider nor the network provider are ready to disclose the required information to the other party as it might reveal business or technical secrets that give them a competitive advantage.

Some results are already presented by WP-D. The goal translation management function described in [4] allows the network provider user to specify the required virtual infrastructure without disclosing how this infrastructure have been determined. This only solves a portion of the problem as the optimal virtual infrastructure might be dependent on the actual topology of the network provider.

Further investigation on how to resolve this issue is still in progress and more results should be available later in the project.

NetInf - OConS interface

NetInf interacts with OConS through the O_{ext} interface defined in [3].

There are mainly two classes of OConS services that NetInf will use. Services in the first class are those that facilitate the transport of IO. That includes for example network coding functions, multi-path and multi-protocol transport. For that class of services, the interactions from the NetInf side will be under the responsibility of the NetInf Transport Service (NTS).

The second class of OConS services can be used by NetInf to establish the network part of the NetInf infrastructure. This could be the case in domains where NetInf and OConS are deployed but CloNe is not.

NetInf interface with lower layers

In portions of the network where OConS is not (yet) deployed, NetInf can use traditional transport services based on IP, Ethernet or new innovative transport mechanisms. The NetInf architecture includes a convergence layer to adapt the NTS to the different potential underlying networks.

CloNe - OConS interface

CloNe relies on the concept of Flash Network Slice (FNS) to establish virtual network infrastructure for the cloud users. The FNS allows to connect computing and storage resources distributed in the network in a single or multiple domains. To establish those FNSs, CloNe can provide its own mechanisms but can also rely on the services offered by OConS such as the WAN Interconnectivity for Virtual Networks.

Since CloNe architecture is modular and defines different services for managing computing, storage or network resources, an alternate solution is to implement the resource management service using OConS as illustrated in Figure 5.2.

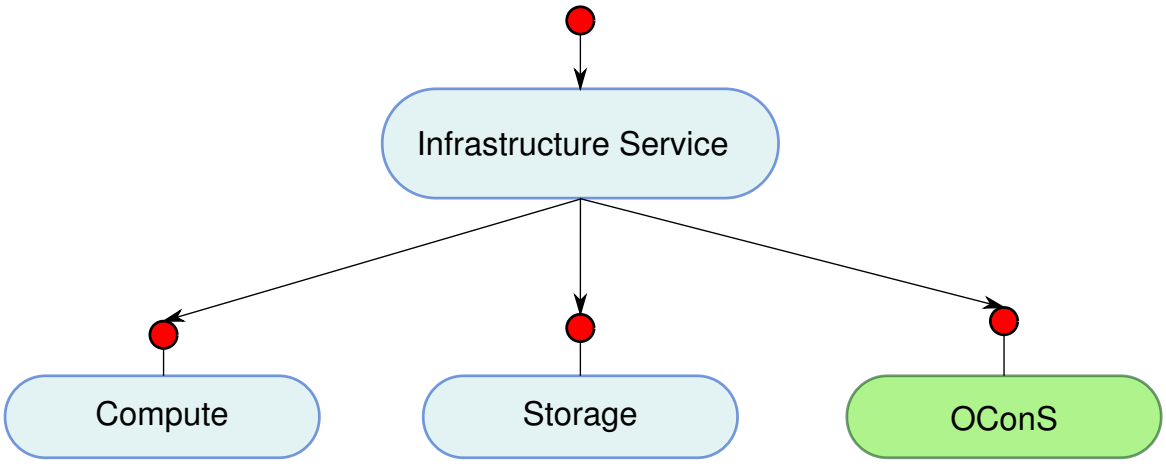

Figure 5.2: Potential interactions between CloNe and OConS

One of the issues identified in the FNS management is the identification of the end-points. Each of the end-points might be in different domains, using different naming schemes (public or private IP addresses, node id, port id or name, ...). Moreover, the end-points might represent physical

or virtual entities (port, node, subnets, ...). There is a need for establishing a naming scheme that allows to unambiguously identify the appropriate end-points. This issue has been partially addressed in the CloNe architecture.

CloNe interface with lower layers

In the absence of OConS, the FNS used by CloNe can be implemented by several means. The CloNe architecture document ([4]) provides a few potential mappings to existing mechanisms, including L2 and L3 Virtual Private Networks (VPNs).

5.2 Theme Interactions

In this section, an overview of initial results from the Themes are presented. The actual description of the Theme work results are (or will be) included in the corresponding work package deliverables.

5.2.1 Security

Amongst others, the architectures identify also the security and privacy issues and how potential related threats are mitigated. According to the approach described in Section 3.2 above, the identified security objectives of NetInf, OConS, and CloNe are as follows.

WP B security objectives: NetInf is involved in developing the SAIL architecture and protocols for an Information Centric Network (ICN) [2]. In ICNs generally, while one is concerned with the usual security goals (confidentiality, integrity, availability/ authentication etc.) as in any other host-based networking scenarios, there are new security considerations that arise. WPB will be investigating (mainly) these new security considerations both at the architectural level and for the specific protocols and prototypes that will be developed as part of SAIL NetInf. In general, in an ICN, one is more interested in content-based security threats and mechanisms, and it is to this that effort in WPB is being devoted.The approach being taken is to iteratively identify threats and potential security services and mechanisms as the development of NetInf proceeds.

The current threat model identifies a number of new ICN-specific threats, for example, “Content Mismatch”, which would occur if an object is requested, but some other object is returned by the network. Various security services (e.g. “Naming Security”, “Content Integrity”) have been defined that could assist in countering this threat, and some specific mechanisms are being examined that can provide this service in an ICN. In this case, “Name/Data integrity” mechanism has been defined that can be used as the basis for both security services. At present, WPB has progress further in the handling of this threat and the related services and mechanisms than for other identified threats.

The set of ICN-specific threats so far identified include:

- Content Mismatch (discussed above),
- Content Snooping, which is a traditional threat, that may be more easily exploitable in ICNs,
- Privacy Invasion, again possibly made worse in an ICN since nodes frequently see entire objects rather than packets,
- False Content Injection, which may be the ICN-equivalent of spam,
- Unauthorised Access, where a bad-actor may access an object intended to be limited to some community,

- Cache Pollution where a bad-actor aims to pollute an ICN content cache, possibly as part of a DoS, and
- Mis-Routing, where a bad-actor influences ICN routing to their benefit.

Work on describing security services that can be used to counter these threats is ongoing, and will have to address a number of difficult mechanism issues, for example, a key management scheme that could scale to the number of objects in an ICN, and some form of authorisation, that may have to be embedded into the objects themselves rather than be provided by the ICN protocols. WPB will of course also need to address how some more traditional host-based security services can be used in an ICN, for example, how to secure cache-update protocols.

WP C security objectives: Main OConS contributions will be in the area of advanced mobility management and transport capabilities in the network [3]. Hence security and privacy related work concentrates on these topics. It addresses misuse prevention and how to ensure system integrity, concentrating on the concerns newly introduced by these added capabilities. For these advanced OConS contributions the security services shall ensure

- the legitimate use of advanced mobility management that support effectively the distributed decision taking and enforcement,
- its misuse prevention and thus the potentially affected availability, and
- accountability of having used such mobility management functionality.

Regarding the controls of the connectivity resources to advance transport capabilities, security services are not specifically advanced but have to show qualities that are not behind state of the art, i.e.

- need to ensure the availability of functions and elements enabling the transport capabilities and
- accountability is highly desirable, although the extent to which privacy concerns are enforced forms a trade off.

WP C privacy concerns: Privacy requirements are tightly linked with security, with the overarching requirements to ensure protection of users' data and enable user control of the level of this protection. Besides, we consider the broader case, which targets protection of data belonging to operators, service providers or any entity related to either the use of the provision of OConS services. Hence, the exposed information shall be adapted and filtered to other entities depending on the particular policies, but still assuring the correctness of that information. It is most unlikely that there will be either one or a unique solution, e.g. if these entities reside in different legal frameworks and privacy becomes another flavour, or if the shared information gets correlated with other data and thus is breaching privacy concerns. In this sense, the privacy related work is intended to highlight potential issues relating to privacy loss within the OConS architecture, rather than propose specific solutions to ensure it.

WP D security objectives and privacy concerns: CloNe [4] focuses on the development of a complete and flexible architecture for Cloud Networking, with flash network slice capabilities, which will operate as a reference model for deploying complex applications over heterogeneous virtualised networks. Concerning CloNe, the security task focuses on the cloud networking security requirements and challenges [33], while also covering the more fundamental cloud computing security aspects.

The following security objectives were identified to be relevant for cloud networking:

- Availability, which covers the availability of the cloud networking services.
- Integrity, which defines the integrity of data, communication and processes throughout the entire lifecycle of the accessed cloud networking services.
- Confidentiality, which includes the confidentiality of data located in the CloNe infrastructure and in communications.
- Authenticity, which requires the authentication of the CloNe infrastructure components (both physical and virtual components) and the actors, viz. infrastructure service users, infrastructure service providers and the resource administrators.
- Non-Repudiation, which proposes that the components and/or the actors cant repudiate their actions at a later stage. This would ensure conformance to the operational policies defined by the law, participating actors or institutions.
- Privacy, which ensures that the policies defined for handling the private information are adhered to.

Current work includes a security function, which is part of the overall suite of management functions to be deployable and manageable at every layer of the CloNe infrastructure, and its integration with additional configurable modules to implement the security policies defined by the participating entities. The security function shall accept the security goals specified by the involved entities at various levels of the architecture, and translate it to the constraints on the underlying resources. The modules include an authentication module to ensure assurance, authentication and an overall trust structure to be implemented throughout the architecture; and an access control policy module which allows the access control policies to be reflected all the way down to the underlying hardware components.

5.2.2 Inter-provider

During this first year, the Inter-provider Theme's main objective was to raise awareness that the SAIL architecture needs to take into account domains. This objective has been achieved in all technical work packages.

WP-B

At this stage of the project, WP-B has the clearest idea of what a domain means in the scope of their work. It has constructs like Multiple Distributed Hash Tables to delimit domains and make them inter-operate.

WP-C

WP-C has embedded the domain concept in its use cases. The Data Centre interconnection use case identifies different operational domains (the data centres themselves, the interconnection network and the access networks) that need to inter-operate. This use case is the main candidate for the definition and implementation of the inter-domain interface. Figure 5.3 shows the interprovider interfaces identified in this use case.

WP-D

WP-D has started its work on inter-provider aspects after realising that the initial single administrative domain approach is not enough for their work. The attempt of defining a single-domain network highlighted the need for inter-provider interfaces with clean definitions in the CloNe architecture. Currently, distributed and centralised resource information exchange mechanisms between

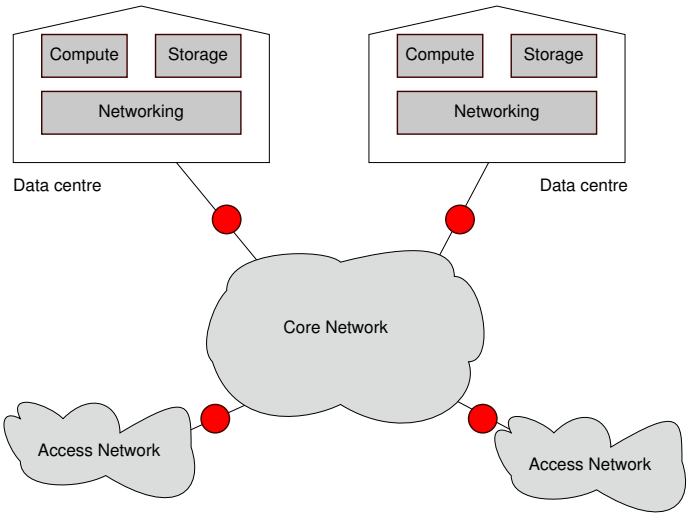

Figure 5.3: Interprovider interfaces in the OConS data centre interconnection use case

domains are being investigated. Next steps include the definition of the information that needs to be exchanged.

Interactions between WP-C and WP-D

The use case regarding the interconnection of Data Centres using Open Connectivity Services shown in Figure 5.3 provides a starting point for discussions between WP-C and WP-D, regarding common functionalities in the interprovider interface that complements the current work of the interface between these WPs.

5.2.3 Management

In this section we describe the first consolidated structure of a management architecture that applies to all of SAILs partial architectures defined in the individual work packages and that adheres to the guidelines defined in Section 3.3. We have noted previously that the adherence to any or all of the guidelines is not mandatory. We will indicate this in the following using examples in the NetInf and OConS partial architectures.

We develop the management architecture's concepts in two steps. In Section 5.2.3.1, we present the simplified architectural view of each work package that is based on the definitions of each work package as laid down in the corresponding deliverables [2–4]. In Section 5.2.3.2, we show how the three simplified architectures can be combined in terms of management in a wider scope, to indicate how the guideline adherence facilitates a homogeneous overall management architecture.

Note that it is not the ambition of this section to present a consistent architecture of all of SAIL's concepts that go beyond management. Furthermore, as in the case of the guidelines, since the analysis is based on the current architectures of the work packages that are still subject to change, further refinement will naturally be necessary as the SAIL project progresses.

5.2.3.1 WP-Specific Guideline Implementation

In this section we focus on the application of guidelines to individual work packages. The purpose of this section is to provide a more detailed understanding of how the theme's generic management

guidelines can be implemented in each work package, such that each work package is able to assess the management approach in its context for further feedback and refinement.

WP-B: NetInf

Figure 5.4 shows a simplified version of the NetInf architecture based on Figure 3.2 and 3.3 in WP-B's deliverable [2], and how management is integrated according to the theme's guidelines.

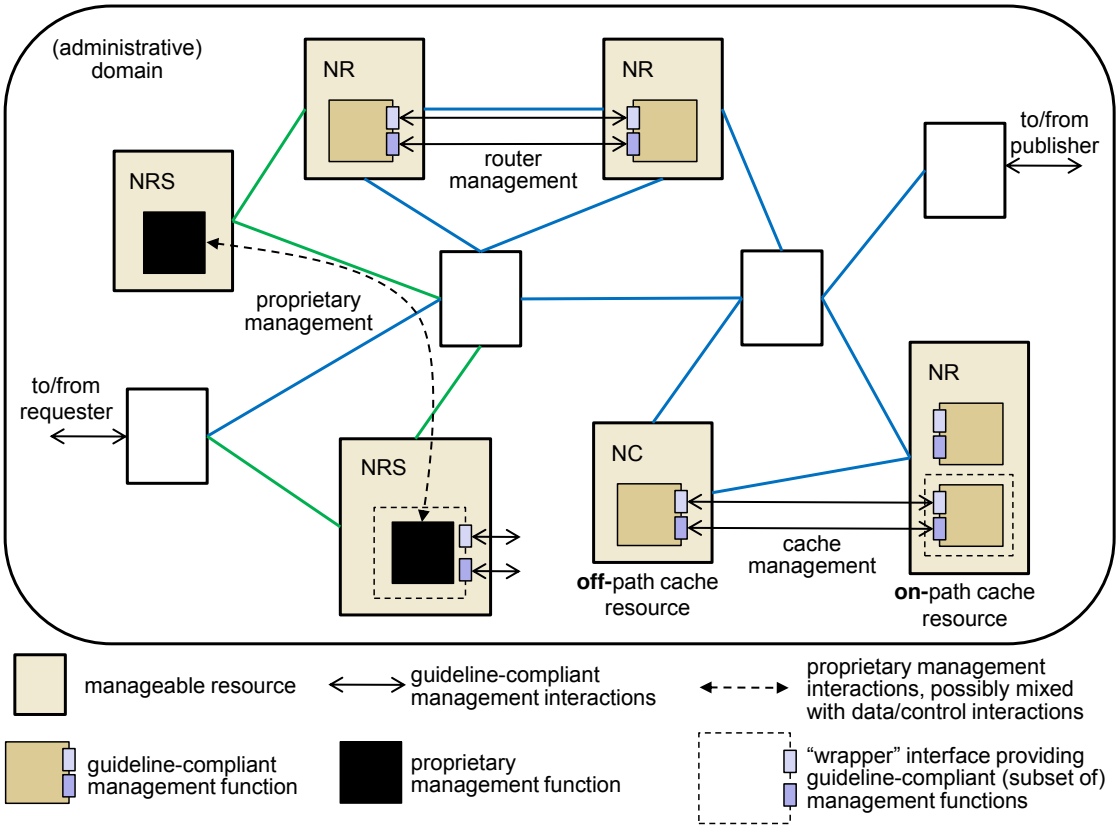

Figure 5.4: Management in the NetInf architecture, based on Figure 3.2 and 3.3 in [2].

In Figure 5.4, we have depicted the main architectural elements of WP-B, including NetInf routers (NR), Name Resolution Service (NRS) elements, and two types of caches: an on-path and an off-path cache, where the former is integrated into an NR, and the latter is a separate architectural element.

According to the theme's guidelines, each architectural element is modelled as a manageable resource, where certain information about the resource relevant for management is provided (we show in Section 5.2.3.2 how that information is fed into WP-D's knowledge plane). Further, each management function and function interactions are modelled either in a proprietary or native form, that is, in compliance with the theme's guidelines. In the case of proprietary management, a "wrapper" interface may be implemented to tap into a subset of the management function's capabilities that are desired to be published to the overall management plane. This situation reflects the first guideline, which states that guidelines are in principle non mandatory. In Figure 5.4, this fact is indicated for the bottom NRS. In the native case, interactions between individual management functions occur according to the two management interfaces defined in WP-D (the controller and collaboration interface, see Figure3.2). Observe that management functions are general embedded

into the network elements, which is partly inspired by the in-network management approach of 4WARD's work package 4 [20] and also the approach supported by WP-D's management concepts. Embedding is often natural, because many functions are inherently distributed. This applies to NetInf for cache management and the management of distributed NRS resources, where distributed embedded management principles directly apply. Figure 5.4 furthermore indicates a single domain, which according to the theme's guidelines denote the context of the architectural elements / resources. Section 5.2.3.2 provides more details on domains as contexts and on how a domain context interfaces with the knowledge plane defined in WP-D.

WP-C: OConS

Figure 5.5 shows the main architectural elements in the OConS architecture, based on Figure 4.2 in WP-C's deliverable [3]. These include the Information Management Entity (IE), Decision Making Entity (DE), and Execution and Enforcement Entity (EE), together with a selection of OConS control interfaces (coloured lines).

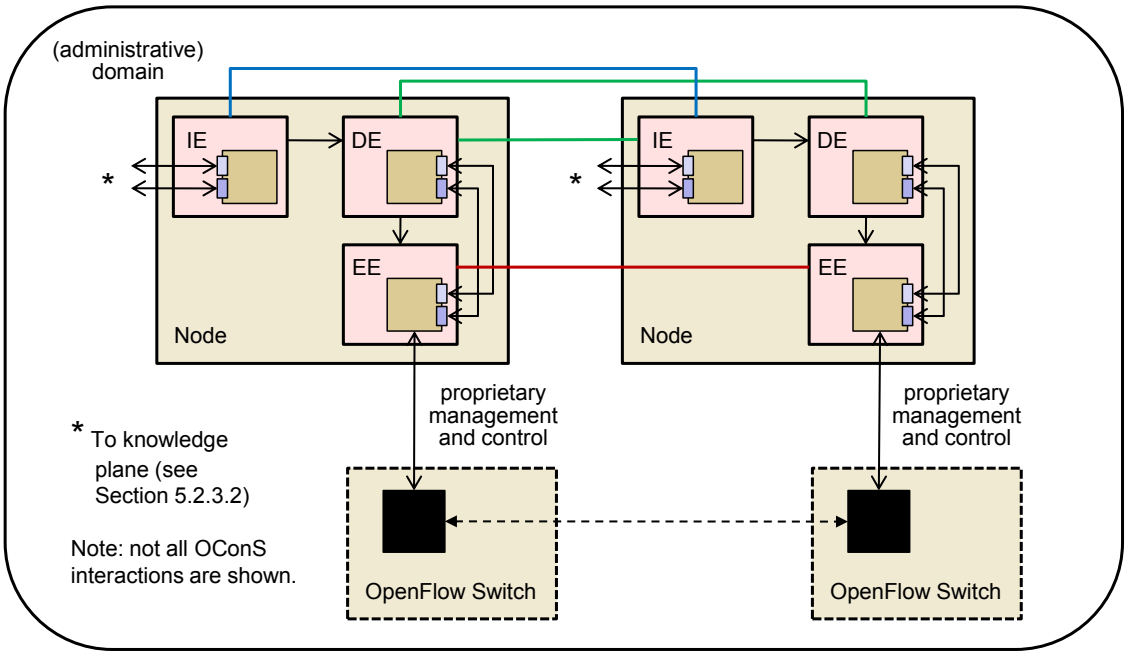

Figure 5.5: Management in the OConS architecture, based on Figure 4.2 in [3]. Not all OConS interactions are shown, for colour codes see Figure 4.2 in [3]. For symbols, refer to Figure 5.4.

Management is integrated into the OConS architecture according to the theme's guidelines much in the same way as for WP-B's NetInf. A difference to NetInf is that in OConS, a node contains multiple entities (IE, DE, EE). While the complete node can be modelled as a manageable resource, it is also viable to model each of the individual entities of the node as a manageable resource. This is primarily a question of granularity, and there is no restriction which granularity is to be chosen. Consider the IE resource in particular. In Figure 5.5, we have indicated by two outward arrows that the IE connects to the knowledge plane (also refer to Section 5.2.3.2). In terms of resources, it makes sense from the viewpoint of the knowledge plane to consider only the IEs as resources of interest, since they contain and manage information which is exactly the purpose of the knowledge plane as discussed in [4]. As a consequence of the chosen granularity of resources in Figure 5.5,

management functions that comply with the theme's guidelines are implemented also on the level of that resource, that is, each entity. Management interactions between entities (that is, manageable resources), then occur also on the same level.

From the discussion of NetInf and OConS, a main observation therefore is that resources, and as a consequence, management functions, may be modelled at any adequate level of detail (nodes, entities, also layers and even domains), but appear homogeneous from the viewpoint of the management framework. This is particularly important because of the fact that WP-D introduces in addition virtualisation concepts, in which this abstraction is also vital when physical and virtual domains need collaborative management.

Figure 5.5 further contains an example of an initial migration step, in which the management of OpenFlow-related resources (that is, OpenFlow switches), is maintained on a proprietary level. This is indicated by the fact that only the EE's management function has a connection to the OpenFlow switch. At a later stage, it is possible to add a guideline-compliant management interface also to the OpenFlow switch that replaces the proprietary interface in order to integrate OpenFlow-related management also in the overall management framework.

WP-D: CloNe

In the scope of the management theme, the cloud networking WP acts as the main source of management concepts. WP-D's deliverable [4] describes these management concepts in the context of flash network slices (FNS) in detail. Due to the many possible architectural configurations in WP-D, encompassing peer-based and hierarchical interactions and mixtures thereof that may be applied in various forms to the data, control, and management plane, we focus in this section on an example configuration to show how the theme's guidelines also apply to WP-D.

Figure 5.6 shows a setup in which two domains each contain two architectural elements (network and storage) to construct a "storage network" flash network slice (FNS). Management interactions follow primarily the peer-to-peer paradigm but can be readily applied to hierarchical patters as well (e.g. for goal translation), but which are not shown in the figure. Figure 5.6 is based on Figure 2.4, 4.4, and 5.1 in [4], and Figure 11 in [22].

Let us first consider the physical domain at the bottom of Figure 5.6. This domain contains physical resources of the substrate from which FNS can be created. In this example, the network node (e.g. an OpenFlow switch) and storage node are both modelled as manageable resources according to the network management theme's guidelines. In the example, this is vital in order for the physical resources to be available for assignment to virtual resources in the virtual domain for constructing flash network slices (FNS).

The network node and the storage node each contain the three types of management functions from Figure 5.1 in [4]. This way, both types of nodes can be controlled in terms of Distributed Goal Translation (DGT), Distributed Resource Management (DRM), and Distributed Fault Management (DFM). Interactions between any pair of these components is according to the two types of interfaces defined in WP-D and captured by the theme's guidelines. Similar as in Figure 5.5, the DRM and the DFM both also communicate with the knowledge plane (not shown in this figure). On node level, both network node and storage node communicate also via the two types of management interfaces. On domain level, the physical domain may communicate with other physical domains (shown in more detail in Section 5.2.3.2).

Consider now the virtual domain, in which network management functions (DGT, DRM, DFM) are arranged in the very same way as in the physical domain. This is possible because conceptually, virtual and physical resources are identical. In the example, the two virtual resources shown make up a flash network slice, which may correspond to a "storage network". Compliance with the guidelines shows that it is straightforward to set up management-related communication between the physical and virtual domain, which corresponds to inter-domain management interaction. This

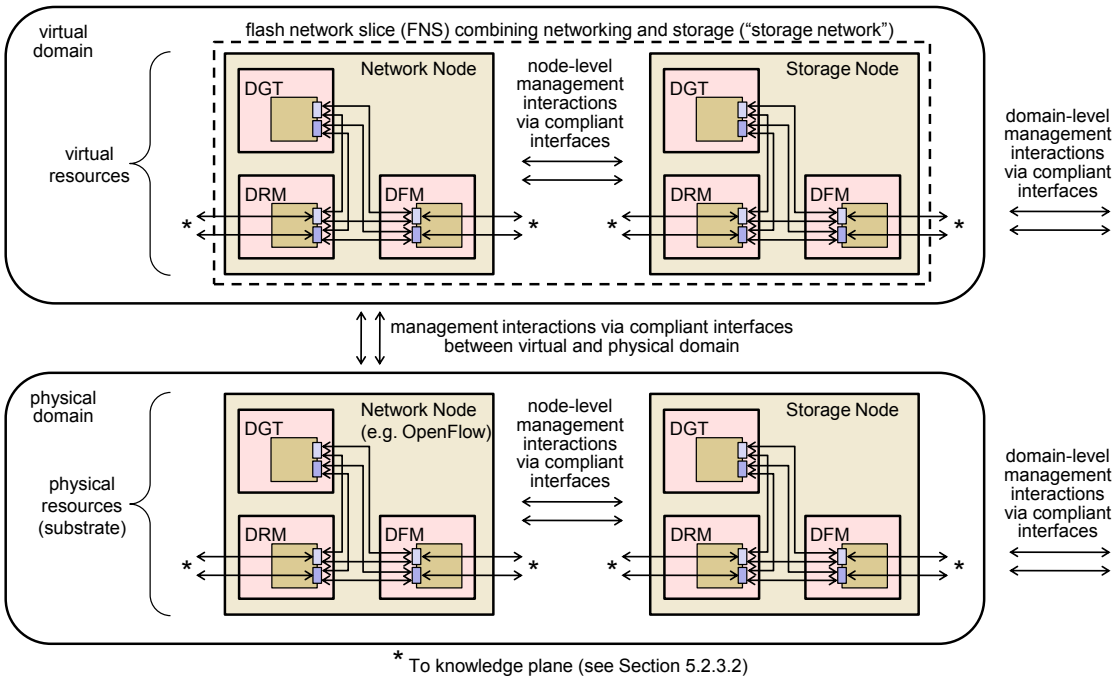

Figure 5.6: Management in the CloNe architecture, based on Figure 2.4 and 4.4 in [4]. The figure shows only an exemplary setup in the peer-to-peer interaction case, among many other possibilities.

is possible because all management-related interfaces follow Figure 3.2 and can interact directly. Note that the semantics of the interaction is not defined at this stage and is out of the scope of the theme's work, but must be sensibly composed by the algorithms defined in each of the work packages.

5.2.3.2 WP-Overarching Guideline Implementation

In the previous sections, we have shown how the network management theme's guidelines can help in the establishment on homogeneous management architectures in each of the technical work packages WP-B, WP-C, and WP-D. In the following, we show how in a broader scope, WP-specific architectures are able to interact in terms of management. As noted, we do not claim the interaction between WP architectures in more general terms than management. Nevertheless, this section may serve as a valuable starting point also for discussions towards how to achieve a SAIL-wide architecture that combines all WPs' sub-architectures not only in terms of management but also on the level of data plane, control plane, layering, etc.

In Figure 5.7, we illustrate inter-WP management relations for a number of typical scenarios, rather than providing an overall view of a complex system that combines all three technical work packages' architectures. It is then possible to combine these with the results from Section 5.2.3.1 into more complex setups that involve all WPs and their interrelations in terms of management.

Figure 5.7.a illustrates the interaction in terms of management between different functional layers. We use the example of NetInf and OConS, based on Figure 3.5 in WP-B's deliverable [2]. As noted before, management functions can be integrated at different levels of detail. In the example, functions are shared on the level of a network element in the NetInf layer (NetInf router, NR), on the level of a protocol layer (convergence layer), which may be implemented as part of the

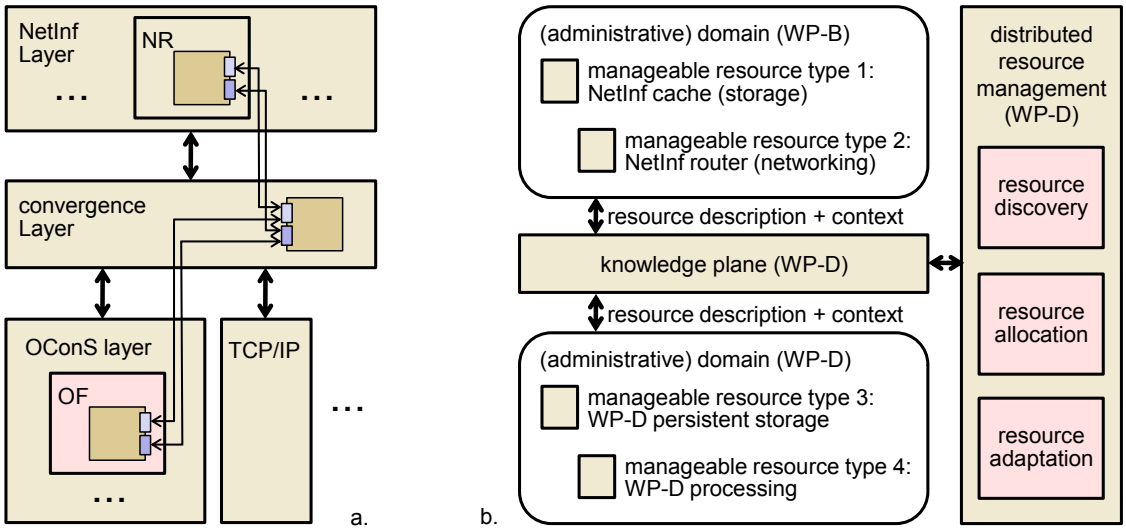

Figure 5.7: Inter-WP management interactions. — a. Management across layers, based on Figure 3.5 in [2] and Figure 4.2 in [3] (NR: NetInf router; OF: orchestration function (OConS)). — b. Types of manageable resources and domains as resource contexts and interactions with the knowledge plane, based on Figure 5.1 and 5.3 in [4]

network stack of some types of network elements, and on the level of a specific function (OConS Orchestration Function (OF)), which is part of a layer. All interactions between the management functions are mediated via the controller and collaboration interface (Figure 3.2).

Figure 5.7.b shows how different parts of the SAIL architecture, WP-B and WP-C in the example, interact with the knowledge plane for the purpose of exchanging information about different types of manageable resources and the context of these resources. In the top administrative domain, WP-B's resources are shared with the knowledge plane introduced in WP-D (compare to Figure 5.3 in [4]). These include, but are not limited to, NetInf caches and NetInf routers. In the bottom domain, WP-D shares information about its resources also with the knowledge plane, including persistent storage resources and processing resources (e.g. from cloud-based servers). Each resource is described homogeneously according to the theme's guidelines.

We have chosen the above example to show how this leads to a number of synergies that different WP's can in turn exploit in terms of management. Consider for example the NetInf router, which may require additional processing resources to perform content-based forwarding. The processing resource provided by WP-D may be appropriate for this purpose. Consider another example where WP-D aims to extend storage resources to include short-term, cache-based storage in the network. For this purpose, the NetInf cache resources may be appropriate. In both examples, WP-D's distributed resource management functions, shown on the right side of Figure 5.7.b, can be employed to determine such configurations and allocate the appropriate resources on behalf of different parts of the SAIL architecture, such as in WP-B and WP-D.

Figure 5.7.b also shows how information about the resources contexts can be shared, which can be used to determine e.g. during resource allocation which resources are eligible for a certain purpose at all. It is important to note that such context is beyond the description of individual resources and describes additional restrictions on how these resources may be used. Such restrictions may include domain-specific properties, for example, the maximum amount of a server's processing capacity that a domain may allocate to any other domain, or the maximum fraction of an individual NetInf cache that can be allocated for other purposes than NetInf internal information management. All such

context is also shared with the knowledge plane and becomes available for the distributed resource management functions provided by WP-D.

5.2.4 Migration

The Standardisation and Migration task is not yet started at this stage of the project. Results from this task will be included in a separate deliverable due at the end of the project.

6 Conclusion

In this document we put in place a set of principles and guidelines to support the architectural work being done in SAIL.

In this first phase of the project, architectural work has been progressing well. The three technical WPs have established a first draft of their architecture. A first harmonised architecture has been drafted putting the WP in relation. We succeeded to identify the interfaces between the different work packages so the WPs can integrate well in an overall architecture. The loose coupling between the WPs enables a smooth migration path as it allows to deploy each WP separately.

The interfaces between WPs still need to be further refined. Some problems in the interfaces between the work package have been identified and we are investigating potential solutions to resolve them. Specific cross work package sessions are already included in the project agenda.

The applicability of the architecture to the overall project scenario and use cases defined in [25] has still to be evaluated.

From the Theme perspective, each of them have enumerated their specific objectives and approach and provided a comprehensive set of guidelines. This has been achieved despite the fact that in the current project organisation, no resources are specifically dedicated to the Theme work. In this first phase of the project, the efforts from Theme representatives in the WP were mainly focused on resolving the issues in their specific work package, rather than contributing to the overall harmonisation of the approaches.

It is seen a need that the Future Internet needs *"to be secure, both to protect privacy and freedom of information, and to minimise abuse"* [34]. This is well considered in the SAIL project. Achievable protection goals have been selected, security guidelines collected as best practice and security objectives were carefully defined. Up to this stage, the approach taken to define development directions, proves that security is taken seriously in the context of available resources.

The management theme has defined a set of guidelines that facilitate the integration of SAIL-wide components in terms of management on the level of individual resources, the resources' contexts (e.g. the domain they are located in), management functions, and data modelling. Detailed examples applying to individual WPs on one side and to inter-WP relations on the other side illustrate how those guidelines can be implemented based on WP-D's management concepts to achieve an overall management architecture based on the decentralised self-management paradigm. Further refinement of both the guidelines and architectural details is to be expected naturally as SAIL's individual architectures progress further.

All WPs have a good grasp on the functions/services that are provided by a domain. Progress is being made on identifying the information that needs to be exchanged over the interfaces between the domains.

We have defined validation framework that will allow an easier integration of the multiple components prototyped. Prototyping and experimentation effort will also help to validate the architecture and clarify the WP internal and external interfaces.

Now that the architectural bases are in place, some effort will be dedicated to increase the coordination and harmonisation of the architecture. The architecture work will keep progressing during the remaining part of the project and a more complete architecture will be reported at the end of the project.


SAIL

Bibliography

[1] The SAIL project web site. `http://www.sail-project.eu/`.

[2] Petteri Pöyhönen, Ove Strandberg, et al. The Network of Information, Architecture and Applications. Deliverable FP7-ICT-2009-5-257448-SAIL/D.B.1, SAIL project, July 2011. Available online from http://www.sail-project.eu.

[3] SAIL. Architectural Concepts of Connectivity Services. Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.1, SAIL project, July 2011. Available online from http://www.sail-project.eu.

[4] Paul Murray (ed.). Cloud Networking Architecture Description. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.1, SAIL project, July 2011. Available online from http://www.sail-project.eu.

[5] Guideline, Free Online Dictionary. `http://www.thefreedictionary.com/guideline`. Last seen on 2011-07-25.

[6] Principle, Wikipedia definition. `http://en.wikipedia.org/wiki/Principle`. Last seen on 2011-05.

[7] Principle, Free Online Dictionary. `http://www.thefreedictionary.com/principle`. Last seen on 2011-07.

[8] The Open Group. TOGAF Version 9 – an Open Group Standard. `http://pubs.opengroup.org/architecture/togaf9-doc/arch/`. Last seen on 2011-07-20.

[9] Representational State Transfer, Wikipedia definition. `http://en.wikipedia.org/wiki/Representational_State_Transfer`, May 2011. Last seen on 2011-07-25.

[10] ISO/IEC 42010 Systems and software engineering — Recommended practice for architectural description of software-intensive systems, July 2007.

[11] R. W. Shirey. Internet Security Glossary, Version 2. `ID draft-shirey-secgloss-v2-08.txt`, November 2006.

[12] User Network Interface (UNI) 1.0 Signaling Specification, 2001.

[13] Architecture principles: Creating the foundation for robust architecture. `http://www.ibm.com/developerworks/library/ar-archprinc`, 2007. Last seen on 2011-07-25.

[14] Microsoft Patterns & Practices Team. *Microsoft Application Architecture Guide*. Microsoft Press, 2009.

[15] B. Carpenter. Architectural Principles of the Internet. RFC 1958 (Informational), June 1996. Updated by RFC 3439.

[16] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational), December 2002.

[17] Future Internet Reference Architecture Group. http://ec.europa.eu/information_society/activities/foi/research/fiarch/index_en.htm, 2011. Last seen on 2011-07-25.

[18] María Ángeles Callejo and Martina Zitterbart. Architectural Framework: new release and first evaluation results. Deliverable FP7-ICT-2007-1-216041/D-2.3.0, 4WARD Project, January 2010. Available online from http://www.4ward-project.eu.

[19] Jukka Salo et al. New Business Models and business dynamics of the future networks. Deliverable FP7-ICT-2009-5-257448-SAIL/D.A.7, SAIL project, July 2011. Available online from http://www.sail-project.eu.

[20] Alberto Gonzalez (ed.). In-network management design. Deliverable FP7-ICT-2007-1-216041-4WARD/D-4.3, 4WARD project, May 2010. Available online from http://www.4ward-project.eu.

[21] Dominique Dudkowski, Marcus Brunner, Giorgio Nunzi, Chiara Mingardi, Chris Foley, Miguel Ponce de Leon, Catalin Meirosu, and Susanne Engberg. Architectural principles and elements of in-network management. In *Mini-Conference Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009)*, Long Island, NY, USA, June 2009.

[22] Dominique Dudkowski, Bioniko Tauhid, Giorgio Nunzi, and Marcus Brunner. A prototype for in-network management in naas-enabled networks. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, Dublin, Ireland, May 2011.

[23] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[24] Geoff Huston. Analyzing the Internet's BGP Routing Table. http://www.cs.columbia.edu/~hgs/internet/bgp.txt. Last seen on 2011-07-25.

[25] Thomas Edwall et al. Description of Project-wide Scenarios and Use Cases. Deliverable FP7-ICT-2009-5-257448-SAIL/D.A.1, SAIL project, February 2011. Available online from http://www.sail-project.eu.

[26] OSI. List of Licenses ordered by Name. Last seen on 2011-07-25 at http://www.opensource.org/licenses/alphabetical.

[27] The GNU General Public License. http://www.gnu.org/licenses/licenses.html#GPL. Last seen on 2011-07-25.

[28] The Current State of FreeBSD and BSD Licenses. http://www.freebsd.org/doc/en_US.ISO8859-1/articles/bsdl-gpl/article.html#CURRENT-BSDL. Last seen on 2011-07-25.

[29] Frequently Asked Questions about the GNU Licenses. http://www.gnu.org/licenses/gpl-faq.html. Last seen on 2011-07-25.

[30] Open Source Initiative OSI – The BSD 3-Clause License. http://www.opensource.org/licenses/BSD-3-Clause. Last seen on 2011-07-25.

[31] Open Source Initiative OSI – The BSD 2-Clause License. http://www.opensource.org/licenses/BSD-2-Clause. Last seen on 2011-07-25.

[32] BSD Licenses at Wikipedia. http://en.wikipedia.org/wiki/BSD_licenses. Last seen on 2011-07-25.

[33] Peter Schoo, Volker Fusenig, Victor Souza, Márcio Melo, Paul Murray, Hervé Debar, Houssem Medhioub, and Djamal Zeghlache. Challenges for cloud networking security. In *Proceedings of the MON-AMI Conference*, Santander, Spain, Sept 2010.

[34] C. Calude. A Dialogue on the Internet with Dr. Brian E. Carpenter, 2008.