D1.1
SUPERCLOUD Architecture Specification

Project number:	643964
Project acronym:	**SUPERCLOUD**
Project title:	User-centric management of security and dependability in clouds of clouds
Project Start Date:	1st February, 2015
Duration:	36 months
Programme:	H2020-ICT-2014-1
Deliverable Type:	Report
Reference Number:	ICT-643964-D1.1/ 1.0
Work Package:	WP1
Due Date:	Nov 2015 - M10
Actual Submission Date:	9th December, 2015
Responsible Organisation:	TUDA
Editor:	Markus Miettinen, Ferdinand Brasser, Ahmad-Reza Sadeghi
Dissemination Level:	PU
Revision:	1.0
Abstract:	We present the architectural design of SUPERCLOUD, a technical framework allowing users of cloud services to deploy ensembles of computational, storage and data communication services transparently over a number of different cloud service providers (CSPs). Such ensembles, so-called *user clouds* or *U-clouds* are strictly isolated from each other and provide fine-grained security self-management facilities. To realize U-clouds, the SUPERCLOUD architecture is divided in three abstraction layers: the compute abstraction plane, the data abstraction plane, and the network abstraction plane. In this document, we describe the overall requirements for the architecture, the sub-architectures realizing the abstraction planes, as well as their interfaces and interconnections and provide a validation of the requirements with regard to two use cases arising from health care scenarios.
Keywords:	architecture, cloud services, multi-cloud, cloud-of-clouds, service virtualization, self-management, security

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 643964.

This work was supported (in part) by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0091.

Editor

Markus Miettinen, Ferdinand Brasser, Ahmad-Reza Sadeghi(TUDA)

Contributors (ordered according to beneficiary numbers)

Marc Lacoste, Nizar Kheir (ORANGE)
Marko Vukolic (IBM)
Alysson Bessani, Fernando Ramos, Nuno Neves (FCUL)
Majid Sobhani (TUDA)

Disclaimer

Executive Summary

Today, despite apparent benefits achievable through locating business processes and data in cloud-based systems, distributed cloud computing raises many security and dependability concerns. This is caused by the increased complexity of such systems and the lack of interoperability between heterogeneous, often proprietary infrastructure technologies. SUPERCLOUD thus proposes new security and dependability infrastructure management paradigms that are : 1) user-centric, enabling self-service clouds-of-clouds where customers define their own protection requirements and can avoid technology and vendor lock-ins; and 2) self-managed, for self-protecting clouds-of-clouds that reduce their administration complexity through automation techniques.

This document presents the high-level architectural framework of the SUPERCLOUD system. We define the actor roles of SUPERCLOUD users and cloud service providers (CSPs) whose services are used to compose ensembles of computation, data storage and data communication services, on which users can run desired computations and store data. We refer to such *user-specific ensembles of computational, data and networking services* as *U-Clouds*. The SUPERCLOUD architecture consists of three abstraction planes: the *compute plane*, the *data plane* and the *networking plane*. The purpose of each of these planes is to provide abstractions of computation, data storage and data communications services used by execution environments that SUPERCLOUD users run on the SUPERCLOUD infrastructure.

A background study is presented which shows the relevant state-of-the-art literature in areas that are central to the technical implementation of SUPERCLOUD. These include works on virtualization and self-management architectures, cloud data management and network virtualization architectures.

We present the common requirements for the SUPERCLOUD architecture derived from use cases related to healthcare that will be developed in WP5. As such use cases represent the highest standard for requirements regarding the privacy of data in civilian applications, we can make sure that solutions developed in SUPERCLOUD will have sufficient means for protection of data privacy also in most other application areas.

Another important aspect of the SUPERCLOUD architecture addressed in this document is the notion of security self-management. The SUPERCLOUD architecture provides a security management framework that connects to each sub-architecture and provides facilities allowing each SUPERCLOUD user to adapt fine-grained, user-specific security settings for data, computation and communications inside her own U-Cloud.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

This document describes the preliminary overall architecture of SUPERCLOUD. It combines and connects the *compute architecture* (developed in WP2), the *data architecture* (developed in WP3), and the *network architecture* (developed in WP4). We first provide an overview of the motivation and vision of SUPERCLOUD before providing a high-level overview of the SUPERCLOUD architecture. We then provide a short survey of relevant state-of-the art: an overview of computing virtualization, cloud data management and network virtualization architectures. We then present the high-level requirements that SUPERCLOUD needs to fulfil in order to be applicable for the intended use-cases, especially in healthcare. We then present the technical details of each of the sub-architectures and the overall architecture, focusing on the high-level design and interconnections between the sub-architectures. Finally, we validate the architectural design by mapping exemplary use cases from the health care domain to show how these can be supported through the facilities provided by the SUPERCLOUD architecture. The results presented in this document should be seen as a description of work in progress. Details of the architecture are subject to change as the work in the project proceeds.

1.1 Motivation

Cloud computing today is very provider-centric. An increasing number of competing cloud service providers (CSPs) operate multiple heterogeneous clouds. Each provider offers its own offerings for customer VMs, with a great diversity of services in areas like security (e.g., system and network intrusion monitoring, firewalling, anti-malware, data security), resource management (e.g., elastic load balancing), or high availability (e.g., checkpointing/recovery). Unfortunately, the adoption of cloud computing services suffers from a lack of homogeneity. Many services are not being deployed due to deficiencies in the possibility for unified control and interoperability of services between service providers. Lack of user control results in vendor lock-in, as services are tightly coupled with their provider and are dependent on the provider's willingness to deploy them. Control is also severely limited by monolithic infrastructures preventing fine-grained cloud customization by the customer. The lack of interoperability stems mainly from the heterogeneity of services, and especially from the fact that service-resources mappings are not compatible across providers.
Cloud infrastructures themselves pose many security challenges, arising, e.g., from the new technologies employed, mainly virtualization. These introduce new services and software components, and with them novel security vulnerabilities. Therefore, current provider-centric cloud systems are faced with three major security challenges:

- Security vulnerabilities in infrastructure layers, as each layer (e.g., customer VMs, cloud provider services, provider hypervisor) is vulnerable to attacks. For instance, a hypervisor and its over-privileged Dom0 is a target of choice for attackers due to its complexity. This makes also devising an integrated protection difficult.
- Lack of flexibility and control for overall security management due to the heterogeneity of security components and policies between cloud providers. This can have a significant security impact as it may introduce vulnerabilities due to mismatching APIs and workflows.
- Security administration challenges. Manual administration of protection of such a highly complex

infrastructure is clearly challenging due to the heterogeneity of its components. Automation of security management would be required, but is lacking in present solutions.

The SUPERCLOUD vision targets the definition and development of a Virtual Private Cloud (VPC) infrastructure similar to now well-accepted Virtual Private Networks (VPN) technologies. For improving interoperability and control, a strong push has recently been made towards a user-centric vision of the cloud.
To lift the interoperability barrier between different CSPs, a resource distribution plane called the SUPERCLOUD is introduced, spanning multiple cloud providers and aiming at decoupling resource production (by cloud providers) from consumption (by users).
To tackle the control challenge, this SUPERCLOUD plane acts also as an extensibility layer, where each new stakeholder operating the distribution layer may provide its own services, enabling the customer to deploy self-service, customizable clouds ranging from Software-as-a-Service (SaaS) to full Infrastructure-as-a-Service (IaaS).
The goal of the SUPERCLOUD solution is thus to provide an interoperability layer between different cloud service providers and propose new security and dependability infrastructure management paradigms that are:

1. *User-centric*: enabling self-service clouds-of-clouds where customers define their own protection requirements and the users are potentially distributed over several different CSPs in order to avoid undesirable lock-ins with a single cloud vendor;

2. *Self-managed*: i.e., self-protecting clouds-of-clouds aim to minimize administration complexity through appropriate automation solutions.

To ensure these goals, the technical architecture of SUPERCLOUD has to provide the following properties:

Self-Service Security : Users can specify their own protection requirements and instantiate and manage security and privacy policies corresponding to their requirements autonomously.

Self-Managed Security : The autonomic security management framework operates seamlessly over compute, storage and network layers, and across provider domains to ensure compliance with user-defined security policies.

End-to-End Security : The architecture provides trust models and security mechanisms that enable composition of services and trust statements across different administrative domains, e.g., across different CSPs.

Resilience The system's resource management framework composes provider-agnostic resources in a robust manner using primitives from diverse cloud providers.

The development of the SUPERCLOUD architecture is driven by real-world use cases in the healthcare domain related to the deployment of distributed medical imaging platforms and laboratory information systems.
The Virtual Private Cloud provided by SUPERCLOUD is realized in the form of so-called *user clouds* or *U-Clouds*. U-Clouds are sets of computational, data storage and data communication services in which individual SUPERCLOUD users can run computations and store data. Individual U-Clouds may be implemented on top of several different CSPs, but are guaranteed to be strictly isolated from other each other.

1.2 Roles in SUPERCLOUD

In the SUPERCLOUD architecture we distinguish mainly between user roles and provider roles. In the following, we detail each role more precisely.

1.2.1 SUPERCLOUD User

A *user* in SUPERCLOUD is an entity using SUPERCLOUD services for a specific purpose. In particular, *user* is using the SUPERCLOUD architecture to run Virtual Machines and to store data in the SUPERCLOUD system. The aim of SUPERCLOUD is that technical details about the physical deployment of virtual machines and cloud storage is abstracted away from the User's view. From the User's point of view, the SUPERCLOUD system presents itself as one uniform infrastructure providing computation and storage services. Note that in most cases, the SUPERCLOUD User is not the same as an *end-user* of cloud-based services. The SUPERCLOUD User is more to be seen as the entity that is running services that are utilized by end-users. For example, in the context of healthcare use cases, a typical SUPERCLOUD User could be a hospital running specific services like a laboratory automation system on the SUPERCLOUD infrastructure, whereas end-users would be employees of the hospital using the laboratory automation system.

1.2.2 Cloud Service Providers (CSPs)

Technically, the SUPERCLOUD architecture is implemented on top of *Cloud Service Providers*, i.e., entities that provide the computational, data storage and networking infrastructure necessary for realizing cloud services. In SUPERCLOUD we distinguish between *public* and *private* CSPs.

1.2.2.1 Public Cloud Service Providers

Public CSPs are entities providing commercial cloud services to their customers. Public CSPs are typically big players benefiting from the ability to provide their services at massive scale, which provides cost advantages. Public CSPs provide their services over well-defined high-level service APIs leaving very little opportunity for individual customers to customize details related to the deployment of their cloud service instances. However, public CSPs typically have very large resources, allowing them to deploy services requiring massive computational, storage or networking capacities.

1.2.2.2 Private Cloud Service Providers

Private CSPs are typically entities providing cloud services for a particular organisation. For example, the IT department of a corporation may act as a private CSP for other units of the corporation. A private CSP may be running a dedicated data centre with appropriate computational and storage services for its host organisation. The private CSP has the advantage that the services it provides towards its clients need not be limited to a well-defined narrow service API, but can be tailored to also provide lower-level control about specific deployment details of the computational, storage and networking services to its clients. Due to the higher level of control they can provide, private CSPs play an important role in SUPERCLOUD.

1.3 Structure

After this introductory chapter, we will provide in Chap. 2 an overview of the relevant state-of-the-art related to virtualization, cloud data management and network virtualization, which form the core technologies on which the SUPERCLOUD architecture will be realized. In Chap. 3 we will lay out the core requirements that SUPERCLOUD needs to be able to fulfil in order to respond to the goals set for the SUPERCLOUD vision. The overall architecture abstraction and the individual compute, data and network sub-architectures will be described in Chap. 4 before discussing the SUPERCLOUD overall architecture in Chap. 5. The mapping of the architecture to concrete use cases is discussed in Chap. 6, before providing a conclusion and summary in Chap. 7.

Chapter 2 Background

In this chapter we provide an overview of the relevant state-of-the art in areas that are central to realizing the SUPERCLOUD architecture. This includes aspects related to virtualization (including isolation and trust) and self-management, data management, and network virtualization.

2.1 Virtualized Computation and Self-Management Architectures

Many architectures and technologies are available today for virtualization of computing resources and for self-management of their security in clouds and multi-clouds. We focus on a few real-world technologies for secure and/or interoperable virtualization, isolation and trust management, and techniques for automating protection and enabling user-control in multi-layer, multi-provider cloud infrastructures, including customer Virtual Machines (VMs). A more detailed survey of the state of the art of techniques for secure computation in cloud of clouds may be found in Deliverable D2.1.

2.1.1 Virtualization of Computing Resources

A distributed virtualization infrastructure should reconcile *horizontal security challenges* linked with multi-provider interoperability with *vertical protection challenges* due to vulnerable software layers. This means fulfilling *vertical design requirements* (e.g., isolation between VM / containers, minimal trusted computing base (TCB) size, user control over architecture and data, minimal overhead, etc.), highly influenced by the abstraction level of the virtualization computing interface. It also means meeting *horizontal design requirements* (interoperability, compatibility with legacy platforms, location awareness...), linked with the level at which interconnection between cloud providers is achieved.
Several virtualization architectures have been proposed to meet such challenges. At the Infrastructure-as-a-Service (IaaS) level, to overcome traditional hypervisor limitations (inflexible infrastructure, large TCB), micro-hypervisors (e.g., NOVA [58], Microsoft Min-V [45] downsizing Hyper-V) aim at minimality by expelling most their code from the TCB. Component-based hypervisors (e.g., Self-Service Cloud Platform [26]) increase user control by pushing modularity into the virtualization infrastructure. Nested Virtualization enables security [60], interoperability [59], and stronger user-centricity through layering by introducing two levels of virtualization, which may be controlled by the user and by the provider. At Platform-as-a-Serivce (PaaS) level, containers (e.g., Docker) enable both efficiency, interoperability, and scalability through the use of lightweight virtualization. A library OS design (e.g., OSv used in the H2020 MIKELANGELO project [44]) aims at giving user more control by tuning the OS to the application; This also provides high efficiency benefits.
To the best of our knowledge, none of the previous designs fully succeeds in meeting user control, low attack surface, interoperability, and compatibility with legacy technologies. However, the hybrid virtualization architecture proposed in SUPERCLOUD, combining nested virtualization, micro-hypervisor, and component-based hypervisor designs allows to achieve successfully user-controlled trade-offs with the provider, both in terms of interoperability and security for the multi-cloud.

2.1.2 Isolation and Trust Management Architectures

For customer VMs in a multi-cloud, while external threats and techniques for their mitigation are fairly well known, insider threats are an open challenge:

- *Attacks from malicious VMs* co-located with the target VM lead to VM isolation breakout (possibly with hypervisor compromise), and VM confidentiality and integrity violations[1]. Such attacks raise the issue of *isolation* between VMs.
- *Attacks from malicious cloud administrators* threaten customer privacy. The cloud provider and its over-privileged administrative domain may use it to inspect customer VMs and access sensitive data. Such attacks raise the issue of *trust* w.r.t. the cloud platform.

Isolation may be based on *trusted software*. Nested virtualization [60] introduces a thin trusted software layer below the (untrusted) provider hypervisor to guarantee VM security and privacy. Self-Service Cloud Computing (SSC) [26] splits privileges of the management VM into provider- and customer-controlled modules. Crypto-as-a-Service (CaaS) [22] refines SSC with transparent storage encryption and VM life cycle management. Such approaches require attestation based on a a hardware root of trust, may pose efficiency issues, or be subject to run-time attacks.
Isolation may also be based on *cryptographic techniques* to achieve secure computation in a widely untrusted environment, encrypting both algorithms and data. Many schemes were proposed in the field of multi-party computation [25], fully homomorphic encryption [34], or verifiable computation [47]. Although the practicality of such techniques in large scale environments is still yet to be demonstrated, they are certainly part of the overall solution.
Isolation may also be *hardware-based*, code being run in *secure enclaves*. Many mechanisms are available such as the SMM protected mode [16], or Intel TXT/AMD SVM instructions. However, they are generally not well suited to the cloud, as code may not be interrupted nor run in parallel, hampering scalability. The Intel SGX technology [42] is particularly interesting for SUPERCLOUD. It fully encrypts memory within an enclave, completely transparently from all other layers, where untrusted code can be run in parallel. Benefits are to run only unprivileged code, to support multi-core and multi-threaded systems, and to require only the CPU to be trusted.

2.1.3 Self-Management of Security

Different *monitoring solutions* have also been explored to assess the security and safety status of virtualized infrastructures. A first class of solutions explores infrastructure monitoring quite broadly. The approach may be per infrastructure service, e.g., focusing on VM co-location [37], firewall configuration, performance, or resource optimization [46]. It may also be per level at which monitoring is performed, either statically by VM image analysis [55], or dynamically with the help of the hypervisor (VM introspection) [32]. A highly active stream of research explores change analysis either for single VMs (Amazon AWS Config service [15]), or for a full cloud infrastructure [23]. Overall, most solutions remain for single clouds or single software layers. General change analysis for dynamic multi-clouds is still lacking, and will be explored in SUPERCLOUD to provide configuration compliance guarantees. More broadly, for mitigating security complexity, transparent self-protection is required, both cross-layer to overcome layer complexity, and multi-provider to overcome security mechanism heterogeneity. It will be addressed by the SUPERCLOUD self-management architecture.
A second class of solutions explores monitoring for specific security properties. For instance, for information flow, either at PaaS or IaaS levels (IBM Trusted Virtual Domains [18]), with strong links with isolation architectures. Another property is infrastructure integrity, with different techniques for verification at language, hypervisor, or hardware levels [16]. Here again, such techniques have strong relations with trust management solutions surveyed previously (e.g., for remote attestation). Overall, solutions remain very layer specific, or for single providers. This motivates the need for a multi-provider, multi-layer trust management framework, well integrated with the security management architecture. Such framework should also offer flexible support for security to orchestrate different types of security services (e.g., isolation, trust).

[1] Typical examples are bounce attacks due to misconfigured device drivers in the hypervisor, or side-channel attacks due to low-level resource sharing.

Finally, regarding *security policies*, a wide range of access control models have been defined to express authorizations in increasingly distributed and dynamic settings, but without yet a full model for fine-grained multi-cloud access control. Sample policies include contracts to express VM security requirements, capture access control at hypervisor-level [53], network equipment security [52], or data-centered policies [54]. Policies remain highly dependent on software layers or on the resource type. They generally do not address flexible user-to-provider trade-offs over security control (e.g., through security SLA abstractions). This motivates the need for an integrated policy management framework for security, with arbitration and SLA negotiation capabilities.

2.2 Cloud Data Management Architectures

In this section, we review some real-world technologies dedicated to the security of data management systems in the cloud. We focus on some existing products permitting to share encrypted data, to protect the data that are stored on the cloud, on anonymization tools, and on dependability techniques. A detailed overview of state of the art in data management in cloud of clouds can be found in Deliverable D3.1.

2.2.1 Sharing Encrypted Data

Many solutions enable users to store their own data on the cloud (*e.g.* Google Drive, Dropbox). However, most of them do not permit user-centric data encryption. One reason is that that user-centric encryption does not easily permit *data sharing*. There are a few commercial initiatives, but these are mostly not based on advanced cryptographic tools.
The MEGA platform [8] is one of such example. After the closing of Megaupload, the founders have decided to create a new platform for data hosting for which they do not have the possibility to obtain the stored data in clear, whereas it is still possible to share data between customers. They propose a system in which the encryption key is shared among the customers having access to the encrypted data.
A similar technical approach has been proposed by Boxcryptor [2], which proposes an application permitting the customers of the most popular cloud storage service providers (Google Drive, Dropbox, etc.) to continue to store their data but in an encrypted way.
Numerous other commercial products enable to encrypt data on cloud, and thereby put the security of data in the hands of the customers. Most of them make use of key sharing (with the inherent problem of key loss) and of data duplication (with good security but poor flexibility regarding the dynamicity of the data and customers). To the best of our knowledge, no existing commercial product makes use of advanced cryptographic tools such as the ones we propose to use in SUPERCLOUD (except the AlephCloud [1] in the USA, which has been closed recently).

2.2.2 Middleware-Based Encryption

Another way to protect data stored in the cloud is to provide a middleware dedicated to the data encryption and decryption (or tokenization). The data confidentiality is then not verified *w.r.t.* the middleware itself, but *w.r.t.* the rest of the world. The middleware then permits to perform computations over protected data since the resulting information can be close to what can be obtained with non-protected data.
Today, several tools provide such kinds of middleware. The most well-known are Perspecsys [12] and CipherCloud [3], which moreover permit to interact with existing cloud service providers such as Microsoft or Salesforce.

2.2.3 Anonymization Tools

Regarding data anonymization, several commercial products exist that provide "simple" ways to anonymize data, using basic methods such as data substitution or data blurring. This is for example the case for the IBM Optim product [6], but also for Oracle Data Masking Pack [10], Informatica Data Privacy [7], or Data Masker [4].
Identity Mixer (idemix) is an anonymous credential system developed at IBM Research - Zurich that enables strong authentication and privacy at the same time. As of 2015, it is deployed in a demo version on the IBM BlueMix Platform-as-a-Service cloud [13]. With Identity Mixer anonymous credentials, a user can selectively reveal any of the attributes contained in the credential without revealing any of their information whatsoever. Thus, anonymous credentials are a key ingredient to protect one's privacy in an electronic world. With Identity Mixer, users can obtain from an issuer a credential containing all the information the issuer is ready to attest about them. When a user later wants to prove to a service provider a statement about her, she employs Identity Mixer to securely transform the issued credential. The transformed credential will only contain the subset of the attested information that she is willing to disclose. The user can apply this transformation as many times as she wants and still none of credentials can be linked to each other.

2.2.4 Dependability Techniques

Today, almost all major Infrastructure-as-a-Service cloud providers (e.g., Amazon, Google, Microsoft Azure, IBM Softlayer, Rackspace, etc.) offer the option of replication across multiple datacenters. These however rely on proprietary protocols and cannot be used to orchestrate multiple clouds under different administrative domains, i.e., belonging to different cloud providers. Similarly, major storage hardware vendors (IBM, EMC/Dell, NetApp, etc.) offer proprietary hybrid cloud solutions.
No widely adopted open source solution for improving dependability in the context of cloud of clouds exists at the moment. However, there are several open source projects, developed in earlier projects by SUPERCLOUD partners, that are used by many research groups around the world. These include BFT-SMaRt [21], SCFS [20], Depsky [19] and Hybris [29].
Finally, on the end user side of the spectrum, many storage front-ends that synchronize and organize multiple public cloud storage services such as MultCloud [9], Otixo [11], CloudCube Android app and others exist. However, these have no goals of boosting dependability nor are they oriented towards business users.

2.3 Network Virtualization Architectures

A network infrastructure that provides full network virtualization must support arbitrary network topologies and addressing schemes as defined by its users. Long standing virtualization primitives such as VLANs (virtualized L2 domains) or MPLS (virtualized paths) are not enough to provide this form of network virtualization. The reason is that these technologies are anchored on a box-by-box basis configuration and do not scale to today's cloud scenarios. As a consequence, current network provisioning can take months, while computing provisioning takes only minutes [38].
In the past few years this situation has begun to change with Software-Defined Networking and the availability of new tunneling techniques. A number of network virtualization solutions have been proposed, mostly targeting a single multi-tenant datacenter. In this section we present a short survey of this recent work.

2.3.1 Software-Defined Networking

Traditional computer networks are vertically integrated. The control plane (that decides how to handle network traffic) and the data plane (that forwards traffic according to the decisions made by the control plane) are bundled inside the networking devices. As a result, networks are complex and very hard

to manage [17]. To express the desired policies, network operators need to configure each individual network device with low-level tools. This has led to a slow pace for the innovation of networking infrastructure.
Software-Defined Networking (SDN) [40] is a new paradigm that breaks the vertical integration by separating the network's control plane from the data plane. Network switches become simple forwarding devices and the control logic is implemented in a *logically centralized* controller, simplifying policy enforcement. While the controller can be implemented as a distributed system, network applications have access to a centralized programmatic model (a global network view), making it easier to reason about network behavior. A simplified view of this architecture is shown in Figure 2.1.

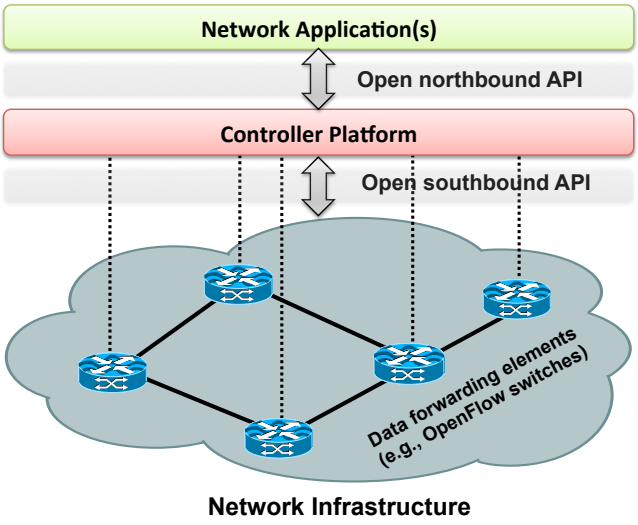

Figure 2.1: Simplified view of an SDN system

The separation of the control plane and the data plane can be realized by means of a well-defined programming interface between the switches and the SDN controller, such as OpenFlow [43]. An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of the traffic and performs certain actions on the packets (dropping, forwarding to specific port(s), modifying headers, among others). Depending on the rules installed by a control application, an OpenFlow switch can – instructed by the controller – behave like a router, switch, firewall, load balancer, or perform other roles.
There is a diverse set of controllers with different design and architectural choices. Centralized controllers include NOX [36] (the first SDN controller) and Floodlight [5]. Naturally, the centralization of control represents a single point of failure and may have scaling limitations. Contrary to a centralized design, distributed controllers such as Onix [39] can be scaled up to meet the requirements of potentially any environment, from small to large-scale networks.
In cloud environments software switches have emerged as the solution for virtualized network infrastructures [56, 27]. The most notable example of a software-based OpenFlow switch implementation is Open vSwitch (OvS) [51]. OvS is a software switch that operates within the hypervisor/management domain and provides connectivity between the virtual machines and the underlying physical interfaces. To support integration into virtual environments OvS exports interfaces for manipulating the forwarding state and managing configuration state at runtime. Since OpenFlow does not allow to modify the switch configurations (e.g. configure queues and tunnels, add/remove ports, create/destroy switches), OvS also maintains a database and exports a configuration interface that enables remote configuration of the virtual switches via the OVSDB protocol [50].
In the following, we give a short overview of two central technologies employed in network virtualization, namely Layer 2 tunneling and network hypervisors.

2.3.2 Layer 2 (L2) Tunneling

Multi-tenant network virtualization platforms [38] typically rely on *tunneling techniques* to implement the needed logical communication between hosts. In these solutions, when a packet is generated in one VM, the tenant's traffic is tunneled across the physical network to the receiving host hypervisor for delivery to the destination VM. Several encapsulation (tunneling) techniques that have been recently introduced to allow Ethernet to span over Layer-3 (L3) networks, allowing multi-tenancy in data centers.
The TRILL (Transparent Interconnection of Lots of Links) IETF standard [49] was proposed to allow a large campus to become a single extended LAN, allowing people to move around. TRILL takes advantage of the best of L2 switching and L3 routing. MAC addressing is used to allow for mobility, while IP routing is used for efficiency. In TRILL a special type of switches, the Routing Bridges (RBridges) [48], encapsulate L2 frames and route them to destination RBridges, which decapsulate them and forward to destination.
GRE, Generic Routing Encapsulation [31], is a generic encapsulation method that allows any protocol to run over any other protocol. Similar to TRILL, Network Virtualization using GRE (NVGRE) [33] uses Ethernet over GRE for L2 connectivity. NVGRE scales to multi-tenant scenarios, allowing 2^{24} tenants (over 16 million) to share the infrastructure (in contrast to VLANs, that restricts the number of tenants to 4096).
Virtual eXtensible Local Area Networks (VXLAN) [41] is an L3 solution developed by VMware used to isolate multiple tenants in a data center. VXLAN is the most common protocol in datacenters for network virtualization and has support from all the major switch hardware vendors. VXLAN is very similar to NVGRE: both solutions offer basically the same functionality.
The Stateless Transport Tunneling Protocol (STT) [28] was developed by Nicira, the SDN-based company acquired by VMware to build this company's multi-tenant network virtualization platform [38]. STT uses Ethernet over a TCP-like stateless protocol over IP. STT has two main advantages. First, it handles large storage blocks of 64kB, solving the problem of efficient transport of large storage blocks in datacenters. Second, it enables hardware offloading for encapsulated traffic with existing Network Interface Cards (NICs).
Currently the IETF is working on a VXLAN extension called Geneve [35], expected to become the future reference. The objective of this protocol is to combine the best of current network virtualization encapsulations (VXLAN, NVGRE, and STT) into a single protocol.

2.3.3 Network Hypervisors

The programmability offered by an SDN system has the means to provide full network virtualization – not only isolation of virtual networks, but also topology and addressing virtualization. FlowVisor [57] was the earliest attempt to virtualize an SDN. This *network hypervisor* acts as a proxy between the controller and the forwarding devices to provide an abstraction layer that slices an OpenFlow data plane, allowing multiple controllers each to control its share (its slice) of a single physical infrastructure. The isolation properties provided by FlowVisor thus allow several networks to co-exist. Five slicing dimensions are considered in FlowVisor: bandwidth, topology, traffic, device CPU, and forwarding tables. Different mechanisms are used to slice over each dimension. For instance, VLAN priority bits are used for bandwidth isolation. In FlowVisor each network slice supports a controller, i.e., multiple SDN controllers can co-exist on top of the same physical network infrastructure. Each controller is allowed to act only on its own network slice.
The FlowVisor slicing approach does not offer full network virtualization. It merely slices the network to be shared among multiple tenants. OpenVirteX [14] (OVX) builds on the design of FlowVisor, acting as a proxy between the controller and the forwarding devices, while providing virtual SDNs through both topology, address, and control function virtualization. All these properties are necessary in multi-tenant environments where virtual networks need to be managed and migrated according to the computing and storage virtual resources. To achieve this, OpenVirteX places edge switches at

the borders of the physical SDN network and re-writes the virtually assigned IP and MAC addresses, which are used by the hosts of each tenant, into disjoint addresses to be used within the physical SDN network. With such approach, the entire flowspace can be provided to each virtual network.
FlowN [30] is another proposal that offers full network virtualization. In this platform, tenants can also specify their own address space, arbitrary topology and control logic. However, whereas FlowVisor can be compared to traditional virtualization technology, FlowN is analogous to container-based virtualization, i.e., it is a lightweight virtualization approach. It is designed to be scalable by allowing a unique shared controller platform to be used for managing multiple domains in a cloud environment. Similarly to OVX, to achieve address space and bandwidth isolation the solution maps between virtual and physical addresses using the edge switches to encapsulate incoming packets with an extra header (in their implementation VLAN tags were used) to each tenant.
Contrary to these hypervisors that virtualize an SDN, VMware's Network Virtualization Platform [38] provides full virtualization in data centers without requiring SDN-based hardware (the only requirement is that all servers are virtualized). Tenants' applications are provided with an API to manage their virtual networks. NVP network hypervisor translates the tenants' configurations and requirements into low-level instruction sets to be installed on the forwarding devices. For this purpose, the platform uses a cluster of SDN controllers to manipulate the forwarding tables of the Open vSwitches in the host's hypervisor. Forwarding decisions are therefore made exclusively on the network edge. In order to virtualize the physical network, NVP creates logical datapaths, i.e., overlay tunnels, between the source and destination OvSs. After a forwarding decision is made, the packets are tunneled over the physical network to the receiving host hypervisor (the physical network sees nothing but ordinary IP packets).
Contrary to the SUPERCLOUD network virtualization platform, all recently proposed work in this area target a single-cloud scenario. In addition, the aforementioned works either do not consider security and dependability in their design, or they do so in a very limited manner. In SUPERCLOUD we propose to improve the resilience of the network virtualization platform in a multi-cloud scenario.

2.4 Conclusion

To the best of our knowledge, none of the previous designs fully succeeds in addressing all SUPERCLOUD requirements regarding, e.g., user control, interoperability, security and compatibility with legacy technologies. Therefore, SUPERCLOUD targets a hybrid virtualization architecture that combines nested virtualization, micro-hypervisors, and state-of-the art network virtualization technologies that allow for flexible but efficient user-centric solutions both in terms of interoperability and security for the multi-cloud. Moreover, in current cloud-based data storage solutions no existing commercial products make use of advanced cryptographic tools for data confidentiality protection. In SUPERCLOUD also such novel cryptography-based solutions will be intensively explored.

Chapter 3 Requirements Analysis

In this chapter we present the requirements towards the SUPERCLOUD system, based on the requirements derived the use cases that SUPERCLOUD addresses. We first introduce the methodology used in requirements gathering before discussing in detail specific requirements.

3.1 Methodology

The requirements analysis is based on the initial use cases defined by project partners PH HC and MAXDATA related to healthcare. The use cases are refined and developed in detail in the forthcoming work performed by WP5 and will be elaborated in deliverable D5.1. The use cases are related to the operation of a distributed medical imaging platform and laboratory information systems utilizing a cloud-based deployment model. These use cases provide a good basis for the analysis, since the privacy requirements related to medical data are very strict. The processed data include sensitive patient information that are awarded special legal protection. By focusing on use cases with very high requirements with regards to data privacy and security, we can ensure that the analysis is based on the most demanding set of requirements for handling of user data by cloud solutions.
The requirements analysis is done from two complementary viewpoints: a user-centric view, in which the focus is on the self-defined configuration and protection requirements of users, and, a provider-centric view, where the focus is on the self-protection and self-manageability of the deployed solution in order to ensure integrity and availability of the provided cloud services.
In the following presentation, we follow key words as defined in RFC 2119 [24] to indicate the requirement levels of individual requirements.

3.2 Requirements Overall Architecture

In this section, we describe the requirements related to functionality, reliability, security and manageability that are applicable to the overall architecture, based on the needs of the healthcare use cases derived in WP5. These use cases will be elaborated in detail in Deliverable D5.1.

3.2.1 Functional Requirements

3.2.1.1 Provider and Platform Independence

The primary requirement of the architecture is to provide a Distributed Cloud Infrastructure (DCI) that users can use to deploy cloud applications and services in their specific cloud service instance, a so-called *user cloud* or *U-Cloud*, in a transparent and user-configurable manner. A U-Cloud is an ensemble of computational, storage and data communication services in which users can instantiate and run computations and store data in strict isolation to other users' U-clouds. The DCI SHALL be provider- and platform-independent, i.e., the individual platform solutions employed by individual Cloud Service Providers (CSP) SHALL NOT limit the ways in which U-Clouds can be deployed in the DCI.

3.2.1.2 Isolation

The architecture MUST provide isolation between the U-Clouds of individual users on two levels. On the first level the computation, data storage and network traffic of individual U-clouds MUST be strictly separated. On the second level, the deployments of the U-Clouds MUST be separated from each other so that the behaviour of individual (potentially misbehaving) U-Clouds do not adversely impact the performance of other unrelated U-Clouds, e.g., by consuming all available computational, network or storage resources.

3.2.2 Reliability Requirements

3.2.2.1 Integrity and Completeness of Data

In use cases related to the processing of medical information, there are high requirements on the reliability and completeness of processed data. Especially in health care systems it is of vital importance that all patient data are correct and accessible in their entirety to the health care professionals, as these data are used to make decisions about the treatment of patients and can have far-reaching consequences for individual patients. The SUPERCLOUD architecture MUST therefore guarantee the integrity and completeness of data processed in a U-Cloud.

3.2.2.2 Availability

Use cases may also have high requirements on the availability of services and data. The architecture SHOULD provide appropriate facilities for enforcing such availability requirements on specified services and data assets within a U-Cloud. In particular, the architecture SHOULD provide facilities for specifying and enforcing specific requirements regarding the redundancy, backups and disaster recovery measures for particular data resources as specified in Service Level Agreement (SLA) between the user and CSPs.

3.2.2.3 Performance

Some use cases may require real-time guarantees on response times for the processing of particular data resources. Performance guarantees may be required for on-line user interaction or for interactions with external systems, including physical electronic equipment like automated analysers. The architecture SHOULD therefore provide facilities for guaranteeing predictable response times for specified time-critical computations, data communication as well as storage access times.
In addition, the architecture SHOULD provide facilities for specifying rules on how the system shall degrade or adapt system behaviour gracefully and in a predictable manner in situations in which the CSP cannot deliver the service according to the SLA.

3.2.3 Security Requirements

3.2.3.1 User-Controlled Security Settings

The architecture SHALL provide facilities for users to define fine-grained, self-service security settings for their U-Cloud. Thereby users can control the protection level of their cloud resources.
The possibility for the users to define and implement their own security requirements and solutions poses the need for arbitration between cloud users. The SUPERCLOUD architecture SHALL therefore provide means to prevent misuse of the self-service security capabilities and to support the cloud providers in maintaining the cloud, implemented over all layers and components.

3.2.3.2 Location-Aware Control

The architecture MUST provide facilities for users to control where their data in a U-Cloud physically are stored and processed at a country- or region-specific level. This ability is required to satisfy legal

requirements that may prohibit the transfer of particular types across jurisdictional boundaries. In addition, the user SHOULD be able to actively monitor its allocated resources, including being aware of where his data are processed and stored.

3.2.3.3 Privacy of User Data

The architecture MUST guarantee the privacy of all user data during processing, transit and storage. The privacy of user data MUST be enforced also towards the CSP. The provider SHALL NOT have access to user data without explicit consent from the user.

3.2.3.4 Accountability

The architecture MUST provide adequate facilities for providing detailed audit trails and monitoring for specified data assets and computational resources.

3.2.4 Manageability Requirements

3.2.4.1 Interoperability

The architecture MUST provide infrastructure and platform-level interoperability. It SHALL be able to realize a Distributed Cloud Infrastructure (DCI) with similar flexibility and control as for a single-provider scenario. Usage or migration of resources belonging to different providers SHOULD be achieved with the same flexibility and level of control as for a single provider.
In addition, queries and operations on data MUST NOT be limited to single CSPs. Queries over collections of separate clouds SHALL be possible in accordance with specified security and privacy rules.

3.2.4.2 Legacy support

To support legacy applications and management tools, the architecture SHOULD provide facilities for building DCIs that can incorporate legacy systems in the cloud infrastructure.

Chapter 4 Compute, Data and Network Architecture

In this chapter we introduce the overall architecture abstraction of SUPERCLOUD and describe the structure of the sub-architectures that it is composed of. Details of the overall architecture and the interconnections between the sub-architectures are described in Chapter 5.

4.1 Architecture Abstraction

The purpose of the overall SUPERCLOUD architecture is to allow individual SUPERCLOUD users to instantiate user-specific *U-Clouds* operating on the SUPERCLOUD infrastructure. U-Clouds are strictly isolated from each other and abstract the physical deployment of computational nodes, data storage and data communications inside the U-Cloud across different CSPs. In order to enable these abstractions, the architecture is split into three sub-architectures as depicted in Fig. 4.1, each addressing a particular aspect of the SUPERCLOUD system:

Compute Sub-Architecture Provides the abstraction of a computation plane that is necessary for SUPERCLOUD users to instantiate computational nodes regardless of the actual physical servers performing the computations.

Data Sub-Architecture Provides the necessary abstraction of a data plane for instantiating abstract cloud data storage regardless of the actual providers and data resources providing the physical storage space.

Network Sub-Architecture Provides the necessary abstraction of a network plane for establishing connectivity between computational and data storage resources regardless of the actual networking infrastructure providing the physical connectivity.

In addition, the sub-architectures are supported by a **security management framework** providing SUPERCLOUD users fine-grained control over any computational, data and networking resources that they instantiate through the SUPERCLOUD system.
This approach for the SUPERCLOUD architecture was chosen in order to minimize the complexity of the interfaces between the functional sub-architectures and to be able to clearly define the interdependences between the individual architectural components.

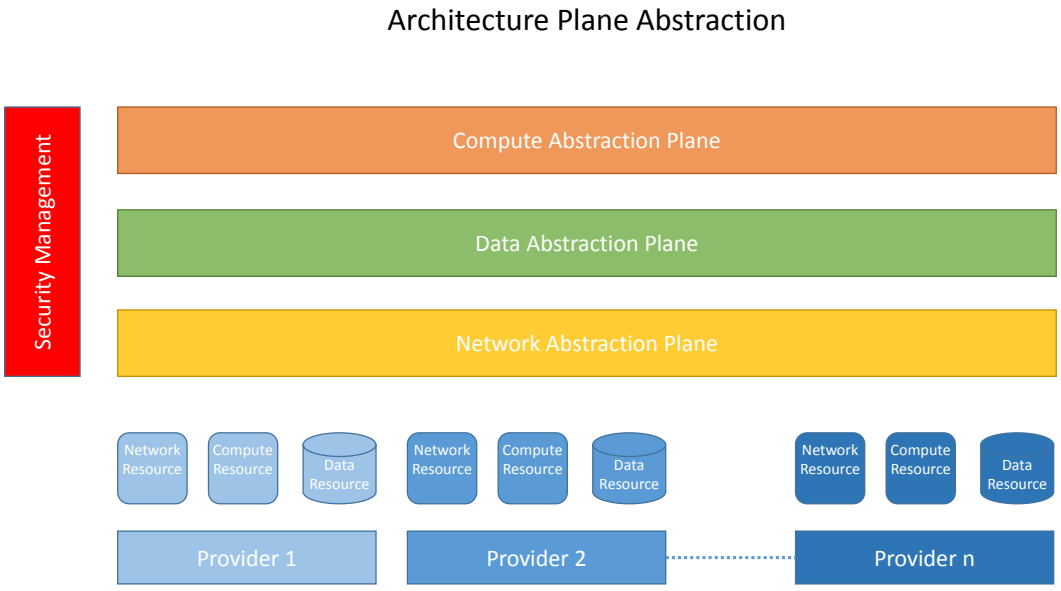

Figure 4.1: SUPERCLOUD architecture

The compute abstraction plane corresponds to the Architecture for Secure Computation Infrastructure and Self-Management of VM Security described in Deliverable D2.1. The Architecture for data management described in Deliverable D3.1 maps to the data abstraction plane, and the network abstraction plane is represented by the Preliminary Architecture of the Multi-Cloud Network Virtualization Infrastructure detailed in Deliverable D4.1. The preliminary structure of the security management framework is described in detail in Deliverable D1.2, the Self-Management of Security Specification. The compute, data and network planes are realized with resources provided by Cloud Service Providers (CSPs). These can be either private or public CSPs (cf. Sect. 1.2.2).

4.2 Functional SUPERCLOUD Architecture

The requirements identified in Chapter 3 build the basis for the *functional architecture* of SUPERCLOUD. The functional architecture describes at a high level what functional properties SUPERCLOUD provides without explaining how this can be achieved. How the architecture can be realized is described in the *deployment architecture* which is described in Chapter 5.

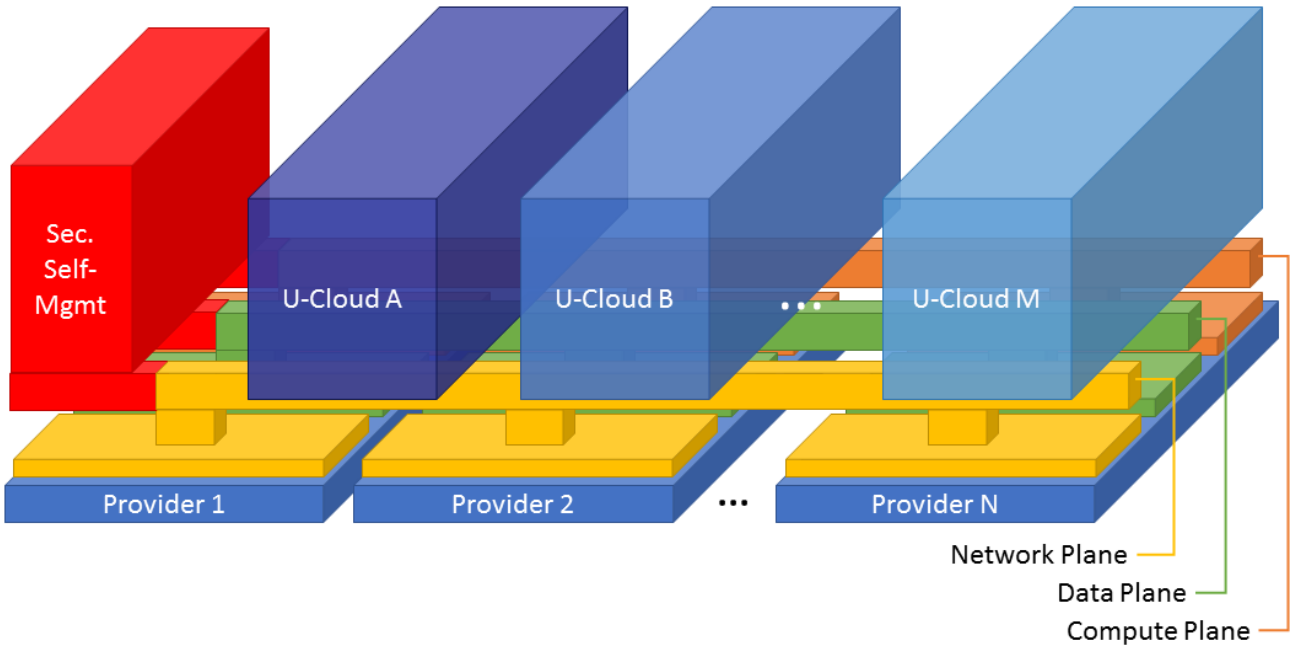

Figure 4.2: Functional view on the SUPERCLOUD architecture

Figure 4.2 shows a functional view on the SUPERCLOUD architecture. The central functionalities of SUPERCLOUD are to abstract the computational, storage and networking cloud resources of (multiple) cloud providers towards the *user clouds* (U-Clouds). Furthermore, the security self-management is a central objective of SUPERCLOUD.
In Figure 4.2 the main components of the SUPERCLOUD architecture are shown. Cloud providers are shown at the bottom of the figure. These cloud providers can be public clouds (like Amazon EC2 or Microsoft Azure) and private clouds (e.g., hosted by the IT department of a private enterprise, serving other units of the corporation).
The resources provided by the cloud providers are abstracted by the abstraction planes and allow U-Clouds to span transparently multiple cloud providers. Furthermore, they provide integration points for the security self-management facilities of SUPERCLOUD.
The U-Clouds rely on the abstraction planes to use the resources of multiple underlying cloud providers. SUPERCLOUD enables an $m - to - n$ mapping between cloud providers and U-Clouds meaning that m U-Clouds can be executed on an abstraction of the resources of n cloud providers. A single U-Cloud can span multiple cloud providers and multiple U-Clouds can be hosted by the same cloud provider. The security self-management of SUPERCLOUD connects to each of the abstraction planes and provides facilities for SUPERCLOUD users to control and manage the security parameters and functions of their respective U-Clouds. The self-management allows SUPERCLOUD users to interact with the SUPERCLOUD and specify personalized settings with a level of freedom not provided by today's cloud providers.
In the following we give a detailed description for each sub-architecture explaining how they interact with the overall architecture and with other sub-architectures.

4.3 Computing Architecture

This subsection gives an overview of the functionalities and architecture of SUPERCLOUD computing interfaces that can be leveraged by SUPERCLOUD data and network management components.

4.3.1 Computing and Self-Management Features

The computing architecture provides primitives to give user-centric control over computing resources of the multi-cloud composed of the cloud services of different CSPs and to manage their security.

4.3.1.1 Functional Features

U-Clouds run above CSPs in a provider-independent manner. Thus the main function of the SUPERCLOUD computing interfaces is that of a *computing hypervisor*. This should be understood as a distributed virtualization infrastructure for running U-Clouds' computing resources such as VM or containers, leveraging computing resources of the different providers.
This hypervisor will typically span a federation of *public clouds* (e.g., Amazon EC2) with low openness and *private clouds* (e.g., leveraging OpenStack) with higher openness enabling greater user control over the infrastructure (see Figure 4.3). If the underlying provider architecture permits it (e.g., in the private CSP case), the hypervisor is designed to be modular enough to enable customization of protection of U-Clouds deep in the virtualization infrastructure.
Thus, U-Cloud protection may be customized thanks to *User-centric System Services (USS)* that extend user-control of the U-Cloud from the hypervisor into the lower layers of the provider infrastructure. USS may be infrastructure services (checkpointing/recovery, live migration,...) directly customizing the computing behavior of a U-Cloud. They may also be low-level hooks for security management of U-Clouds performed using security features described next.

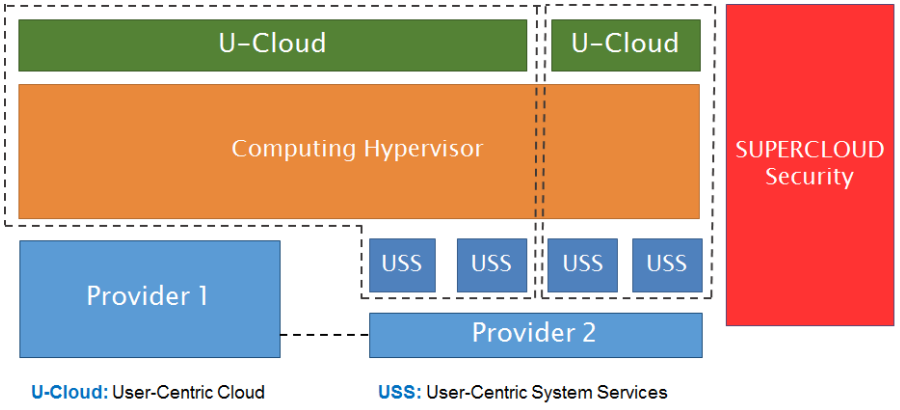

Figure 4.3: SUPERCLOUD computing hypervisor

4.3.1.2 Security Features

A number of security services guarantee the protection of U-Clouds. The idea behind such services is *self-management of security* to implement autonomic security management for the distributed cloud. The security response to a threat is elaborated through orchestration of multiple autonomic security loops.
Viewing self-management of security purely as detection and reaction to threats, two dimensions of orchestration must be distinguished as the virtualized infrastructure is both multi-layered and multi-provider based. *Vertical security orchestration* automates security within and across layers of the distributed cloud, to capture the overall extent of an attack spanning multiple layers and to respond to it. *Horizontal orchestration* provides a homogeneous level of security across clouds, independently from underlying providers, e.g., to detect and mitigate threats propagating from provider to provider. Reaction policies may be distributed and enforced using security mechanisms of underlying providers. More broadly, other security services may be considered. Other non-functional services (e.g., energy efficiency) are also possible. Orchestration then amounts to coordination of per-security service autonomic security loops, possibly composed of other autonomous systems. The security services in the following areas will be considered:

- *Isolation and trust*: different types of *isolation technologies* should be supported, e.g., relying on hardware security mechanisms; *trust management* should also be explored vertically across infrastructure layers, horizontally across providers, and across users.
- *Configuration compliance*: this service is needed for auditability, towards regulatory authorities, or towards customers to increase the level of trust on the security hygiene of the distributed cloud. This component may typically be implemented as an USS.
- *Authorization*: one authorization component is needed to perform resource access control at different levels of the infrastructure.

Some additional components are also needed dealing with policies and orchestration, namely *user-centric* definition of *security policies*, integration relation between vertical and horizontal self-management, and *orchestration* of different security services.

4.3.2 Preliminary Architecture

A simplified view of the logical architecture of the SUPERCLOUD virtualization architecture for computation including self-management features is shown in Figure 4.4.

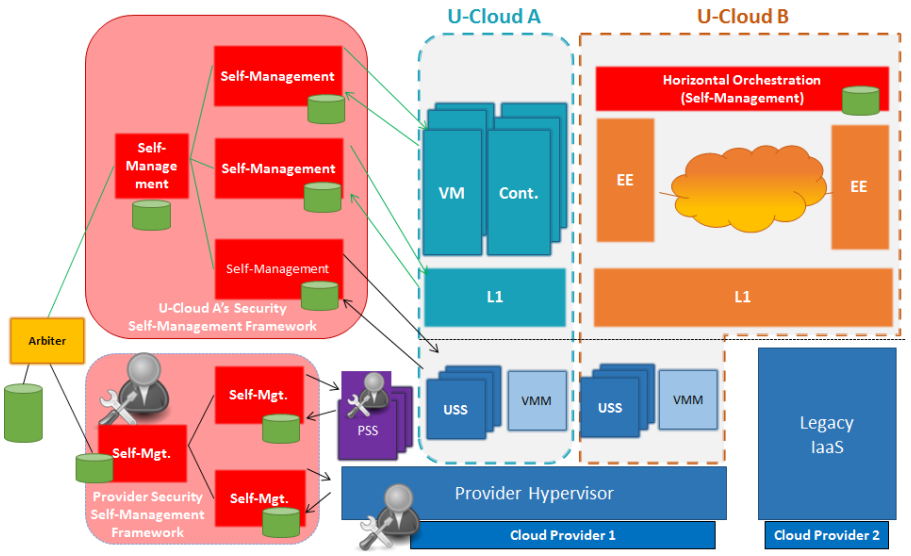

Figure 4.4: Overview of computing virtualization architecture (with self-management features)

Two parts should be distinguished: the *virtualization infrastructure* that provides a distributed abstraction layer for computing resources spanning multiple cloud providers; and the *self-management infrastructure* that implements autonomic security management for the distributed cloud.

4.3.2.1 Virtualization Infrastructure

U-Cloud implementation relies on nested virtualization which offers interesting benefits in terms of both interoperability and security in order to guarantee VM protection in spite of untrusted virtualization layers. Two layers may be distinguished: the *Lower-Layer* (or L0) controlled by the provider, and the *Upper-Layer* (or L1) controlled by users.

- In L1, the user can run *Execution Environments (EE)* which may be Virtual Machines (VMs) (named L2 VMs), or containers – in which case the L1 virtualization platform is a container-platform, Docker style. L2 VMs may be end-user VMs (hosting applications) or management services (e.g., appliances, L1 Dom0). Multiple L1 instances may be deployed across providers, e.g., to migrate EEs transparently. This requires interfacing with the network architecture.

- L0 contains the provider hypervisor and its services. The type of hypervisor may depend on openness of the provider. L0 may be a GPH for a public cloud (e.g., Xen for Amazon). For a private cloud, more modular forms of hypervisors such as a MH may be preferred. This enables clean separation of *provider-controlled services* (PSS) and of a part of L0 under user control. Typically: a *VMM component* for core virtualization functions, e.g., L1 VM creation/deletion; and *infrastructure services* (USS), e.g., live migration.

4.3.2.2 Self-Management Infrastructure

Security self-management is addressed in two aspects:

- *Vertically:* self-management is realized through a hierarchy of Layer Orchestrators (LO) controlling security in each layer: VM/container, L1, and L0. LOs are coordinated by an overall security manager in charge of supervising cross-layer security management. A similar provider-controlled security supervision framework aimed at guaranteeing the protection of its own infrastructure is represented here for simplicity. However, other designs may be possible. An *arbiter* component manages trade-offs between provider and user control over security, such as negotiation over SLAs and security policies. In the design shown, each such hierarchy is U-Cloud-specific to customize self-management per group of customers forming the U-Cloud. Self-management could also be shared among U-Clouds.

- *Horizontally:* self-management is handled by a horizontal orchestration framework, with several possible designs (centralized, fully distributed, hybrid).

A richer description of the architecture of the self-management infrastructure including features for isolation and trust, configuration compliance, authorization, and orchestration of multiple security services is described in Deliverable D1.2.
To summarize, a possible mapping of SUPERCLOUD computing management entities to planes of the overall architecture is shown in Figure 4.5. The figure shows the instantiations of two exemplary U-Clouds (A and B). While U-Cloud B is utilizing only resources from one CSP (Provider 2 in the figure), the instantiations of the computational nodes of U-Cloud A are spanning two different CSP's cloud resources, and are co-ordinated by the Horizontal Orchestration component.
Security Self-Management is realized by the Overall Security Self-Management component that is common for the SUPERCLOUD compute plane and U-Cloud-specific Compute Security Management components that SUPERCLOUD users use to control the security settings that are specific to their U-Cloud. The role of the Overall Security Self-Management component remains then to arbitrate between the security settings of individual U-Clouds and the security requirements of CSPs in order to ensure secure operation of the SUPERCLOUD compute plane as a whole.

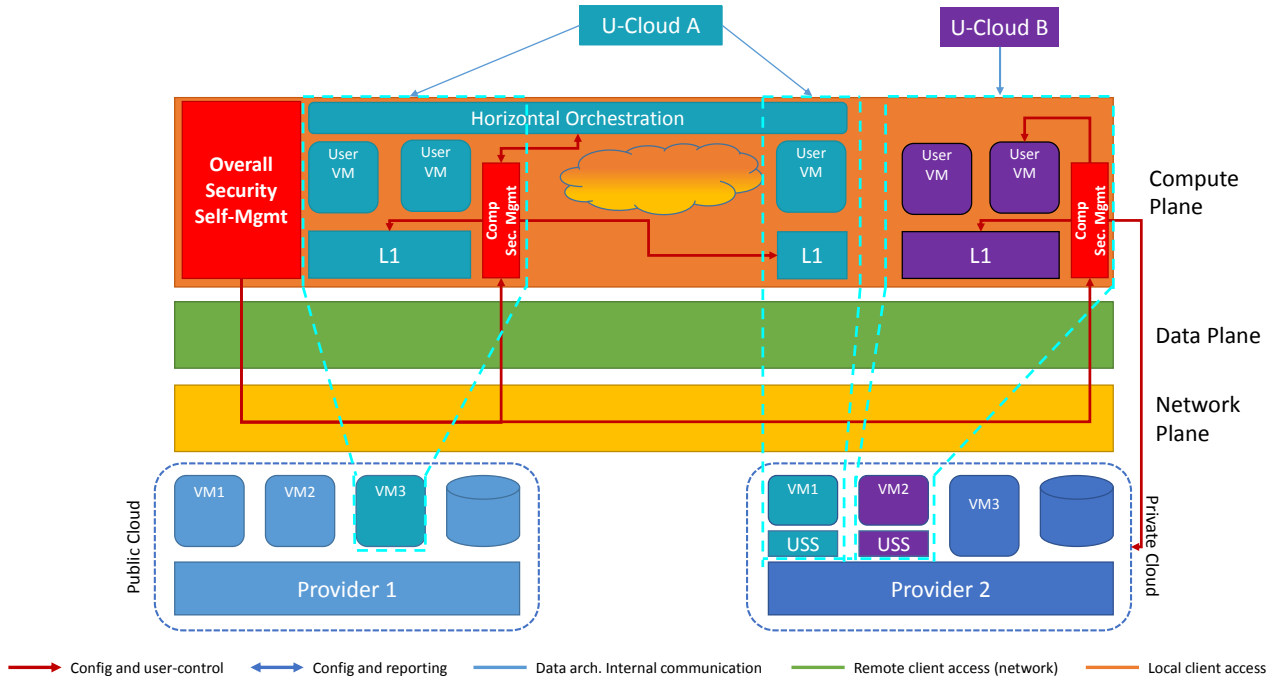

Figure 4.5: SUPERCLOUD compute architecture mapping to the overall architecture

4.4 Data Management Architecture

4.4.1 Data Management Features

In this subsection we overview functionalities and features that SUPERCLOUD data and storage management interfaces will offer to SUPERCLOUD compute nodes.

4.4.1.1 Functional Features

SUPERCLOUD data and storage management interfaces will offer to SUPERCLOUD virtual machines and containers all those APIs that are traditionally offered by commodity cloud providers. Therefore, SUPERCLOUD data management shall support the following storage interfaces: (1) object store interface, (2) block store interface, (3) file system interface; and (4) data management abstractions that cater for efficient, dependable and secure execution of database engines, although Data Management will not be required to natively offer SQL database interfaces.
Furthermore, the size of available storage should be virtually unlimited as the practically unlimited storage from commodity cloud providers may be used to this end. SUPERCLOUD data management should be capable of handling petabytes of storage, supporting various data formats.
Secure sharing capabilities and seamless VM migration across clouds will be supported. SUPERCLOUD shall remain flexible supporting different user-specific storage and data management policies.

4.4.1.2 Dependability Features

SUPERCLOUD data management layer should accommodate the availability requirements stipulated by the users (compute machines). As a rule of the thumb, we expect 99.99% (four nines) and 99.999% (five nines) of availability to be commonplace.
There is no single fault model that applies to all SUPERCLOUD deployments, so the SUPERCLOUD architecture should accommodate for crash faults in the underlying cloud resources (e.g., L0 cloud services can be unavailable) as well as for Byzantine faults (where L0 cloud services are untrusted),

covering arbitrary faults and intrusions. Network should not be assumed to be synchronous and mechanisms for dealing with network asynchrony and partitions should be put in place.
Consistency and data integrity are of primary concern in SUPERCLOUD. SUPERCLOUD targets sensitive applications that require consistency guarantees stronger than what can be found in typical cloud offerings (e.g., eventual consistency). Therefore, the primary consistency criteria for SUPERCLOUD will be *linearizability*, that gives an illusion of traditional, sequential access to data. However, SUPERCLOUD data layers should remain flexible enough to support weaker consistency models, in response to the explicit demand of a user/administrator, to boost performance. Therefore, SUPERCLOUD should support tunable consistency semantics.

4.4.1.3 Security Features

Data privacy is of primary concern for SUPERCLOUD. SUPERCLOUD should enable and implement several redundant and complementary privacy mechanisms so the end user can choose from such privacy offering a mechanism that best suits its needs. Among other techniques, the data management layer will put in place isolation techniques to ensure that data of different tenants cannot be accessible (unless sharing capabilities are explicitly enabled). Sensitive data shall benefit from anonymization features.
Performance and cost optimizations, such as storage deduplication, shall not lower the security levels (according to the principle that security comes first). Where explicitly requested and acknowledged by the user, such performance and cost optimizations may gracefully degrade security guarantees, while providing clear indications of a tradeoff to the user.
Federated identity management across clouds and secure key management solutions should be supported.

4.4.1.4 Compliance

SUPERCLOUD data management shall implement mechanisms for compliance with data location awareness and control as well as data protection. The user shall be able to control which cloud resources are used to store its data, taking into account their location. The data management architecture should remain flexible and follow a future-proof design to accommodate directives and compliance laws that may impose further restrictions.
Laws and directives of the European Union as well as member and associated countries (incl. Switzerland) shall receive priority in fulfilling compliance requirements.

4.4.1.5 Other Non-Functional Features

SUPERCLOUD data management shall attempt at delivering the best performance to the users while satisfying constraints of other requirements outlined above. In particular, our solution shall deliver solutions with the goal of minimizing latency and maximizing throughput.
The cost of SUPERCLOUD storage solutions will often be at odds with other requirements. The SUPERCLOUD data management layer shall allow flexibility to the users in terms of tradeoffs between cost and other guarantees.

4.4.2 Preliminary Architecture

At the high level, the SUPERCLOUD data management architecture comprises a mixture of aggregated resources at multiple clouds (private or public) and value-added functionalities for increasing dependability and security. As SUPERCLOUD data management adds many different and often complementary functionalities, we aim at an understandable logical architecture that will accommodate different value-added security and dependability features.

In the following, we present the logical architecture of data management in SUPERCLOUD which aims at unifying value-added features. Specific architectures of dependability and security features that comply with the high level logical architecture are discussed in Deliverable D3.1.
A simplified view of the logical architecture of the SUPERCLOUD data management layer is shown in Figure 4.6.

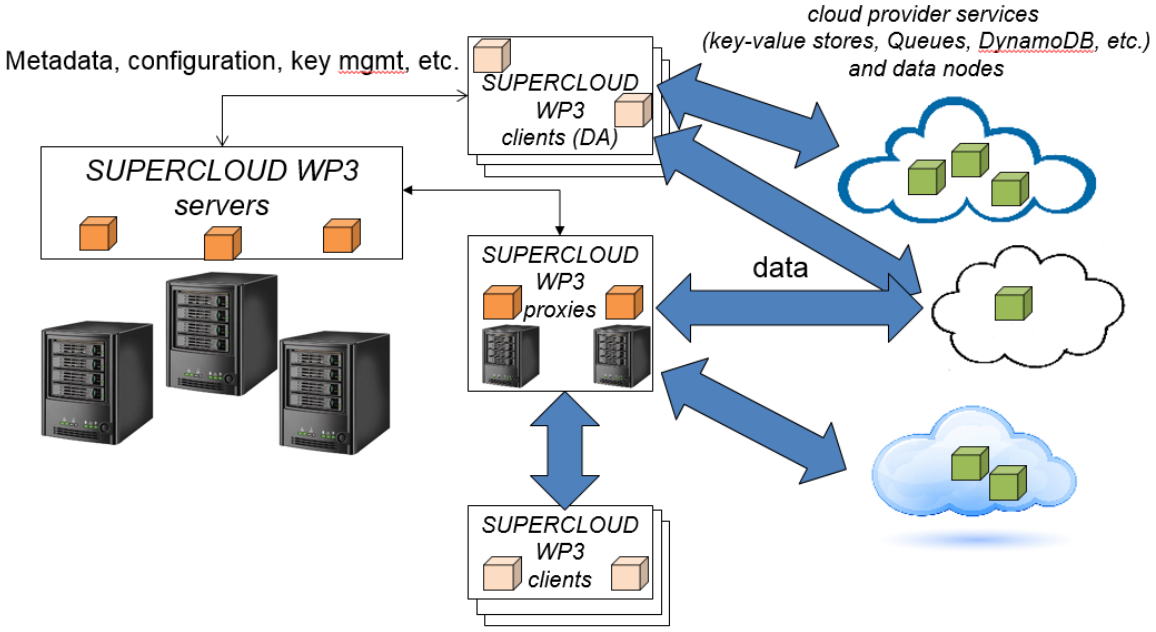

Figure 4.6: High level logical architecture of the SUPERCLOUD data management layer.

Our logical architecture is comprised of five classes of entities:

- *Storage clients* (or simply *clients*) are users of the SUPERCLOUD storage infrastructure. These are all L2 virtual machines (or containers) deployed in the User clouds. Other storage entities may or may not act as storage clients depending on whether they are deployed as L1 virtual machines, in which case they use cloud provider storage or L2 virtual machines in which case they act as storage clients.

  Clients are divided into two subcategories: *direct accessors (DA)* and ordinary clients. Ordinary clients interact with the SUPERCLOUD data management layer via *storage proxy* and are entirely oblivious to the SUPERCLOUD data management layer. This requires minimum changes to clients without installing additional libraries. In contrast, DA clients run SUPERCLOUD specific logic as a client library and can interact and access directly *storage servers* and L1 *cloud provider services*. DA clients can also have certain functionality of storage servers built-in, making them also possibly independent of storage servers.

- *Storage proxies* (or simply *proxies*), are L2 virtual machines (or containers). Proxies are in principle stateless, and can be easily added dynamically to the system. Their primary goal is facilitating clients' access to SUPERCLOUD storage and data management offerings. As we will detail later, examples of proxies include encryption proxies, secure deduplication proxies, etc.

- *Storage servers* (or simply *servers*) are stateful L1 or L2 virtual machines (or containers). The main role of servers is housekeeping of critical portions of metadata vital to operation of the SUPERCLOUD data management layer. Examples of SUPERCLOUD storage servers include storage metadata servers, data integrity servers, configuration management servers, etc.

- *Cloud provider services* (CPS) are L1 cloud storage services that DA clients or proxies can directly access. They expose different APIs, notably object storage and block storage. Examples include Openstack Swift, Amazon S3 and EBS, etc.
- *Cloud provider data nodes* (or simply data nodes) are L1 virtual machines or containers that reside on L1 public or private clouds comprising SUPERCLOUD infrastructure. Complementing CPS, data nodes can perform computation and have locally mounted L1 block storage that ends up storing SUPERCLOUD user data.

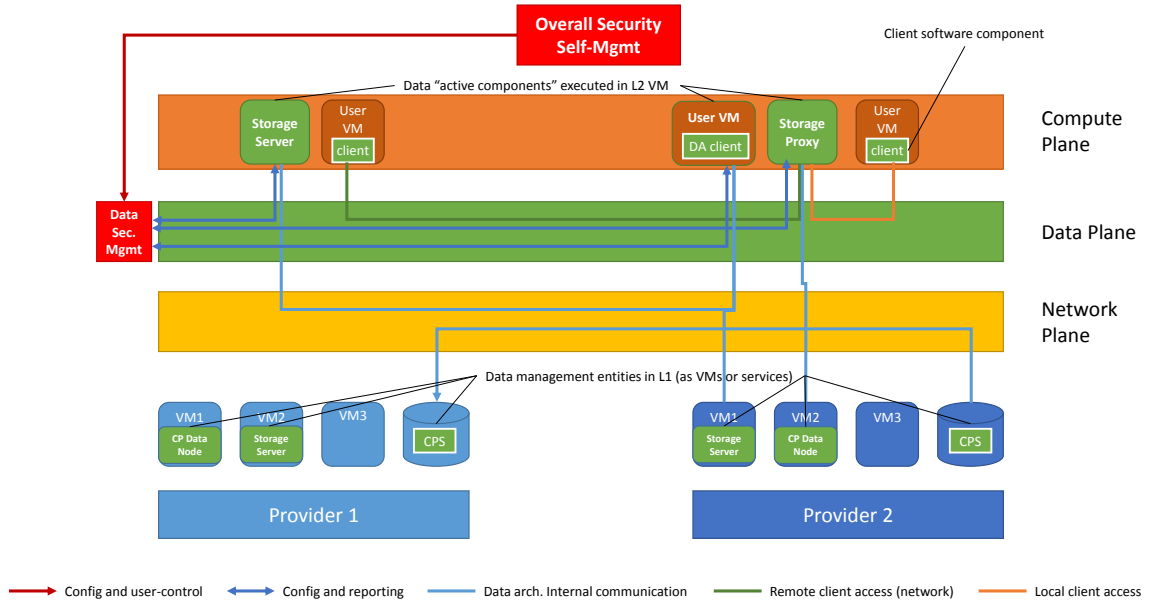

Figure 4.7: Mapping of SUPERCLOUD data management entities to L1 and L2 virtualization layers.

A possible mapping of SUPERCLOUD storage and data management entities to virtualization layers is shown in Figure 4.7. As we can see, storage entities are distributed across L1 and L2 virtualization layers.
Storage clients (Direct Accessors (DA) and ordinary clients) are located in the L2 virtualization layer of the compute plane, whereas storage servers, cloud provider services (CPS) and cloud provider data nodes reside in the L1 virtual machines of the providers' clouds.
Security management of the data management plane is facilitated through the Overall Security Self-Management component, which arbitrates between provider-specified security settings and user-specific settings for their U-Clouds that SUPERCLOUD users can specify with the Data Security Management component that provides the necessary facilities for configuring U-Cloud-specific settings in the data plane.

4.5 Network Architecture

In this section we describe the features of the network virtualization architecture and how they fulfill the network design requirements. We also explain how the network architecture interacts with the overall architecture and with the two other sub-architectures, compute and storage.

4.5.1 Network Virtualization Features

We start by detailing the design principles and features proposed to fulfill the requirements of the network virtualization architecture.

4.5.1.1 Functional Features

The main design requirements that the SUPERCLOUD network virtualization architecture needs to support are network controllability, full network virtualization, and VM snapshotting. Network controllability is fulfilled by having SDN logically-centralized control over the forwarding and configuration state of the software switches that run in the SUPERCLOUD VMs. For this purpose the SUPERCLOUD hypervisor will run Open vSwitch (OvS), the software switch for virtualized environments that is a central part of the SUPERCLOUD solution. OvS exports an interface for fine-grained control of packet forwarding (via OpenFlow) and of switch configuration (via OVSDB). An SDN controller will establish secure connections with each OvS switch to control the forwarding plane. The controller server will also run OVSDB to configure the switches (ports, tunnels, etc.). This allows SDN-based logically centralized control.
The SUPERCLOUD network virtualization platform has also to guarantee isolation between tenants, while enabling them to use their desired addressing schemes and topologies. SUPERCLOUD needs therefore to support full network virtualization. For isolation, the logical centralization of control allows flow rule redefinition at the edge of the network and translation/mapping of physical/virtual events/requests in the logically-centralized network hypervisor. For the physical network to be shared all packets that leave a SUPERCLOUD user VM need to either be tagged (e.g., using VLAN tags or MPLS labels), be re-written, or the packets need to be tunneled. For topology abstraction one possibility is to intercept all LLDP (Link Layer Discovery Protocol) messages[1]. By intercepting all topology-related messages the SDN controller can offer arbitrary virtual topologies to SUPERCLOUD users.
Informed by the network hypervisor solutions surveyed in Section 2.3.3, we have opted for a container-like solution to offer full network virtualization to SUPERCLOUD users. The fact that it is a lightweight solution offers better perspectives in terms of scaling. Also, as security and dependability are core requirements of our architecture, a reduction of the complexity of the necessary distributed protocols is an important advantage.
The third required functional feature of the architecture is support for VM snapshotting, an important management tool used in public and private clouds. We will leverage on well-proven techniques for VM snapshotting and extend them for SUPERCLOUD multi-cloud setting. In addition, we plan to snapshot not only the VM, but also the network state. The motivation is the fact that a VM rarely acts alone, and snapshotting a single VM in isolation would mean losing a significant portion of the complete state that pertains a service.

4.5.1.2 Security and Dependability Features

Security and dependability are core features of the SUPERCLOUD design. We will therefore include resilience as a fundamental requirement of the network architecture. A resilient network virtualization platform requires resilience at both the data and the control planes.
We consider several techniques for the resilient data plane: multipath routing, network coding, and (secure) path monitoring. For the control plane the controller itself, as the core component of the virtualization platform, needs to be resilient. The SUPERCLOUD controller will be fault-tolerant, and techniques to assure strong guarantees of consistency and correctness will be explored. This is important as the consistency of both data and control planes (and their interaction) will be necessary to avoid network anomalies (loops or/and security breaches).

4.5.1.3 Other Non-Functional Features

A virtual network should appear, to its users, as if it were a non-shared infrastructure. As such, the unavoidable overhead of the SUPERCLOUD network virtualization should be low, and the platform should be scalable.

[1]LLDP is the protocol used in OpenFlow networks for network discovery.

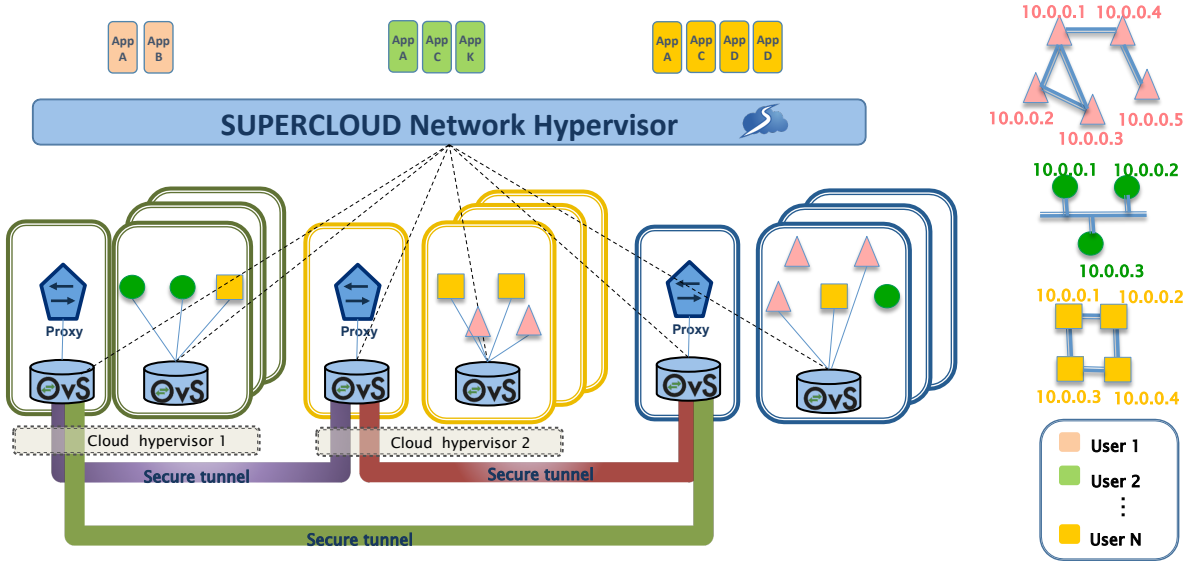

Figure 4.8: Preliminary network virtualization platform architecture

To increase the scalability and performance of the network virtualization platform several techniques will be explored. First, to avoid it being a bottleneck, the SUPERCLOUD controller will be distributed. As distribution brings with it network state consistency challenges, we aim to explore techniques to guarantee consistency and correctness between the different controller instances. In particular, we will explore the interaction of the network component with SUPERCLOUD data and storage. By keeping network state in SUPERCLOUD storage we can leverage on the dependability and consistency guarantees offered by the data management layer.
To improve performance we will further explore efficient algorithms for virtual network embedding and techniques to allow fast virtual/physical mapping.

4.5.2 Preliminary Architecture

The preliminary SUPERCLOUD network virtualization platform architecture is shown in Figure 4.8. The SUPERCLOUD network hypervisor controls and configures the OvS switches that are installed in all VMs (along with the OpenFlow hardware switches, not shown in the figure). This hypervisor is built as an application that runs in the SUPERCLOUD SDN controller, therefore following a container-like approach. Each cloud will have a specific VM that will establish secure tunnels with all other clouds. We call this VM the network proxy. In a distributed configuration the proxy will possibly host an instance of the SDN controller. For each tenant a specific set of network applications that control the tenants virtual network will run on top the SUPERCLOUD network hypervisor, which is responsible for mapping the virtual and physical resources.
To summarize, the mapping of the SUPERCLOUD network virtualization architecture to the overall framework is shown in Figure 4.9. It shows the placement of the OvS virtual switches in the L2 user VMs in the compute plane and in the L1 provider VMs. It also shows the placement of the SDN Controller and Network Proxies in the provider's L1 VMs. Security management is facilitated through the interplay of the Overall Security Self-Management component and the Network Security Management component, which SUPERCLOUD users use to specify their user-specific settings for the network configurations inside their respective U-Clouds. The Overall Self-Security Management component's role is to arbitrate between these user-specific settings and the providers' network-related security settings in order to ensure security of the overall network plane.

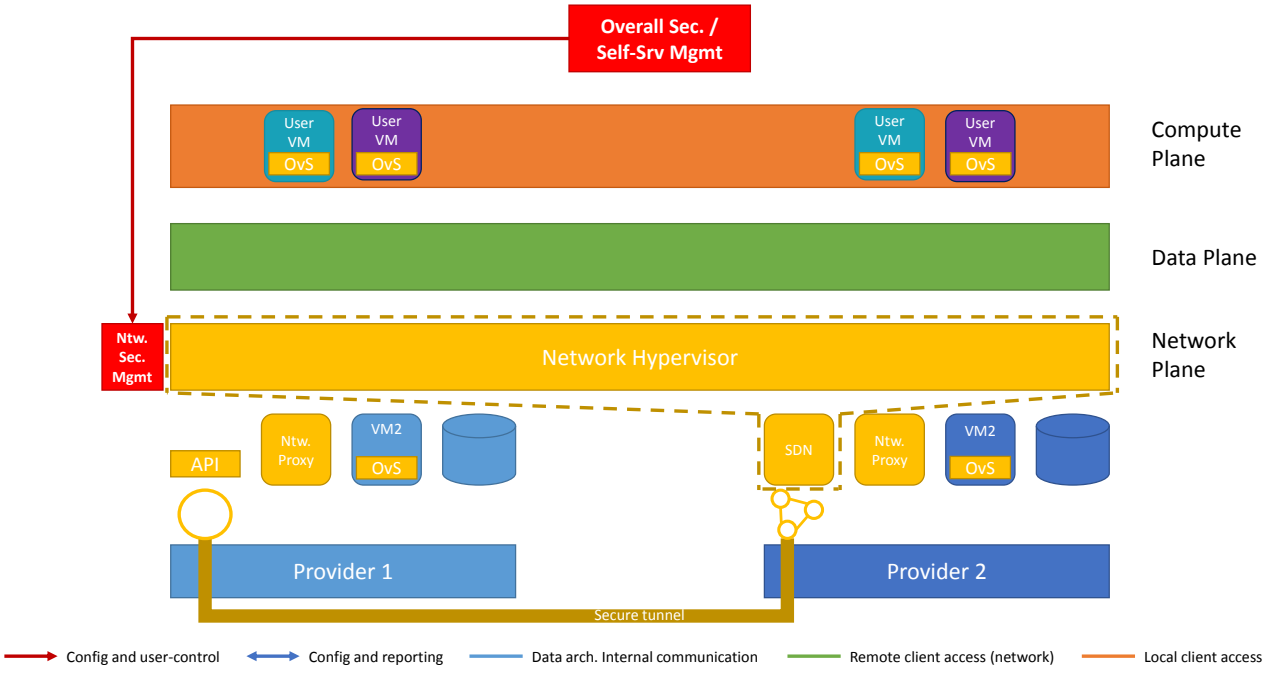

Figure 4.9: SUPERCLOUD network architecture mapped to overall architecture

Chapter 5 SUPERCLOUD Overall Architecture

This chapter describes the preliminary overall architecture of SUPERCLOUD. The architecture is designed based on the explicit requirements derived from the requirements analysis provided in Chapter 3. Implicit requirements stem from the design of the sub-architectures described in Chapter 4. The overall SUPERCLOUD architecture combines the compute, data and network sub-architectures and defines the interconnections between them.

5.1 Structure and Architecture Description Methodology

This Chapter describes the interactions of the sub-architectures with each other in the overall architecture of SUPERCLOUD. The architecture is developed using a top-down approach, in which computational, data and networking services – which are referred to as planes – are connected through well-defined interfaces. This provides an easy to understand high-level description of the general framework which will be refined during the progress of the project, while keeping the basic structure of the construction.
For each of the planes representing one of the SUPERCLOUD sub-architectures a typical interaction with the other planes is shown to illustrate their connections. By investigating the interaction between the compute, data and network plane the connection points between the planes are identified, i.e., the interfaces between the planes.
The interfaces between planes are implemented by different components which are assigned to the individual planes. These components are first explained in plane-specific interface descriptions. Then the deployment architecture of SUPERCLOUD containing the main components of each sub-architecture plane (with focus on the interface related components) is explained.
This approach is chosen to keep the development of the sub-architectures independent of each other by keeping only a few and relatively narrow interfaces between the individual sub-architectures of SUPERCLOUD.

5.2 Interfaces Between Compute, Data and Network Architecture

As mentioned before, SUPERCLOUD architecture consists of components needed to support computations, data management and network communications. The architectures of compute, data and network components are described in detail in the preceding chapter; Therefore an abstract view of them in the form of planes is used to describe the overall architecture. Through interfaces, these components interact with each other and the cloud's resources to fulfil the system's requirements and provide expected functionalities. The compute, data and network planes are based on the resources provided by the cloud providers, which can be private and public CSPs. Defining the interfaces between these components will shape and create the overall architecture of SUPERCLOUD. In the following sections, the interactions and connections between the abstraction planes are described in detail.

5.2.1 Compute Architecture Interfaces

The compute abstraction plane (presented as an orange rectangle in Figure 5.1) provides the computation infrastructure of SUPERCLOUD. The compute plane abstracts the compute resources of the cloud service providers by means of nested virtualization. The compute plane encompasses those components responsible for execution and protection of user VMs, and self-management of VM security. In this plane, a user can run multiple instances of *Execution Environments* (EE), e.g. VMs and containers. This is done in a transparent manner across different providers, whether public or private. VMs on the compute abstraction plane run on a hypervisor (L1 hypervisor), which itself is executed on top of the provider's hypervisor and its services (L0 hypervisor). Figure 5.1 shows this connection between user VMs and provider VMs: uVM1 (user VM 1) is residing in VM2 and uVM2 is residing in VM3 of Provider 1; uVM3 and uVM4 are residing in VM2 and VM3 of Provider 2 respectively.

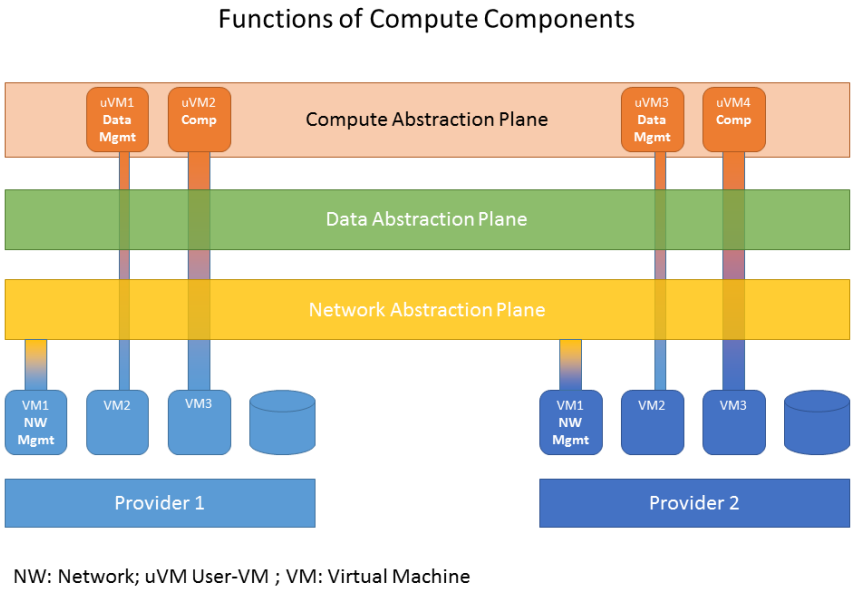

Figure 5.1: Compute architecture interfacing scenario

The compute resources of the compute plane are used for the execution of user-defined tasks. uVM2 and uVM4 in Figure 5.1 provide compute capabilities to the user. However, user VMs are also used to provide compute resources to the other planes. In particular, data management components are executed in user VMs (uVM1 and uVM3 in Figure 5.1).
Although the computation plane provides an abstraction of compute resources it does not eliminate control. For instance, the placement of execution environments is relevant in different scenarios. Figure 5.1 shows that the network management components must be executed for each provider.
The abstraction plane provides the resources to execute the *active* components of the data and network architecture. It further provides control capabilities to meet the specific requirements of the components of the other planes.

5.2.2 Data Architecture Interfaces

The data abstraction plane acts as the storage management interface to the SUPERCLOUD VMs and containers. It provides the data management API for all of the execution environments on the compute plane.
In order to access the cloud's storage, a user's computation VM utilizes data management VM as a proxy to get access to the cloud data. The data management VM runs on the L1 hypervisor and

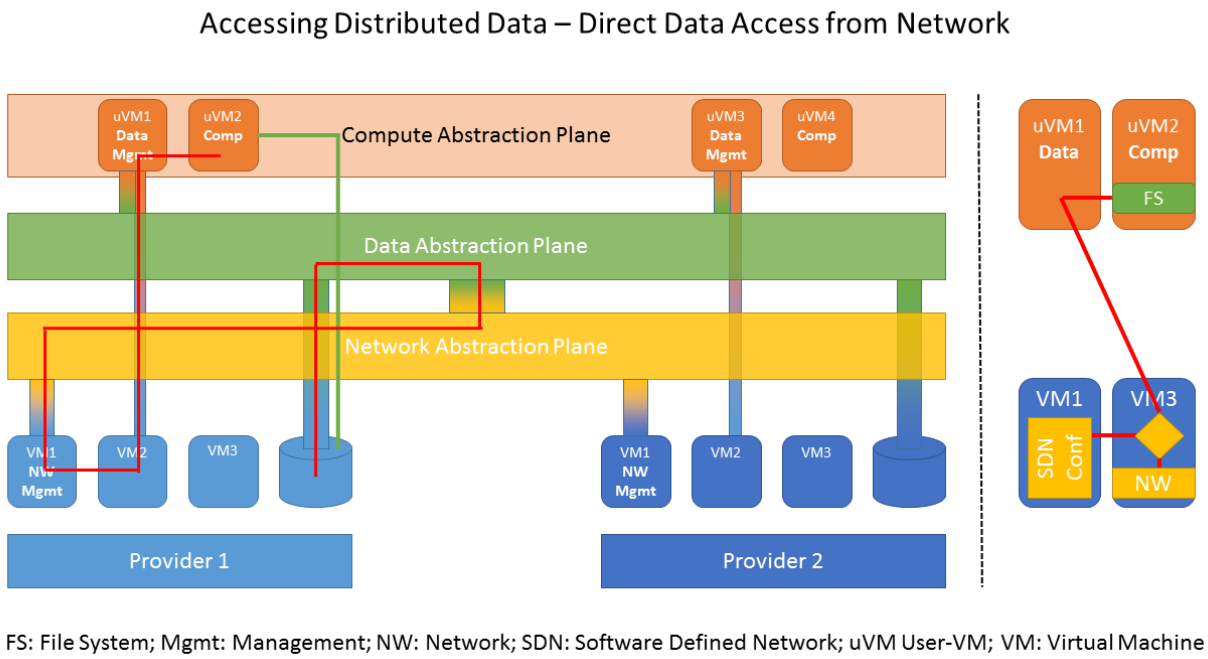

Figure 5.2: Data architecture interfacing scenario

provides APIs for different types of data operations. User VMs relay any data operation request to the data management VM to get access to the cloud's data.
Through the L0 hosting VM on which the data management VM resides, the data operation requests are passed to the cloud's network management VM on the L0 hypervisor. This process is illustrated in Figure 5.2. The hosting VM requests the network management VM to get direct access to the cloud's storage. The network management VM is a network virtualization platform based on Software-Defined Networking [40] (SDN). It separates the network control logic from the underlying routers and switches on the network plane that forward the network requests. Moreover, it provides an abstraction for network topology and traffic routing, as well as isolation between tenants.
On the right side of Figure 5.2 a more detailed view of the interaction between the VMs involved in data accesses is shown. The access origins from an application in uVM2. Inside uVM2 a component (for instance a network file system) redirects accesses to the data management in uVM1. The data management is responsible to guide the data access to the correct destination data store which is accessible over the network. The network plane is then responsible for providing the connection to the data store. In particular, the provider VM2 which is hosting uVM1 has access to the virtualized network provided by the network plane and managed over the SDN controller located in VM1.
The data plane serves as the main interface to the cloud's storage, which is in most cases is a type of Network Attached Storage (NAS). Any requests to access the cloud storage will be forwarded through the network management VM to the data plane. Using SDN virtualization, network management VM routes any data access request to the data plane. Subsequently, it delivers data all the way back to the requesting entity, or performs data operations as requested.
As an alternative, the cloud provider may provide a local storage and not necessarily a NAS, therefore an abstraction needs to be provided to the user VMs that makes the type of storage transparent to them. In this case a provider's L0 VM can directly access the local storage, making this VM responsible for data operations. As a result, the data plane should also provide an abstraction for this type of cloud storage so that the data access remains transparent to user VMs.

5.2.3 Network Architecture Interfaces

In today's cloud network architecture, network logical services must be fully decoupled from the physical network infrastructure. In order to do that, software defined networking (SDN) can be used to separate network control logic from the underlying physical components, e.g. switches and routers.

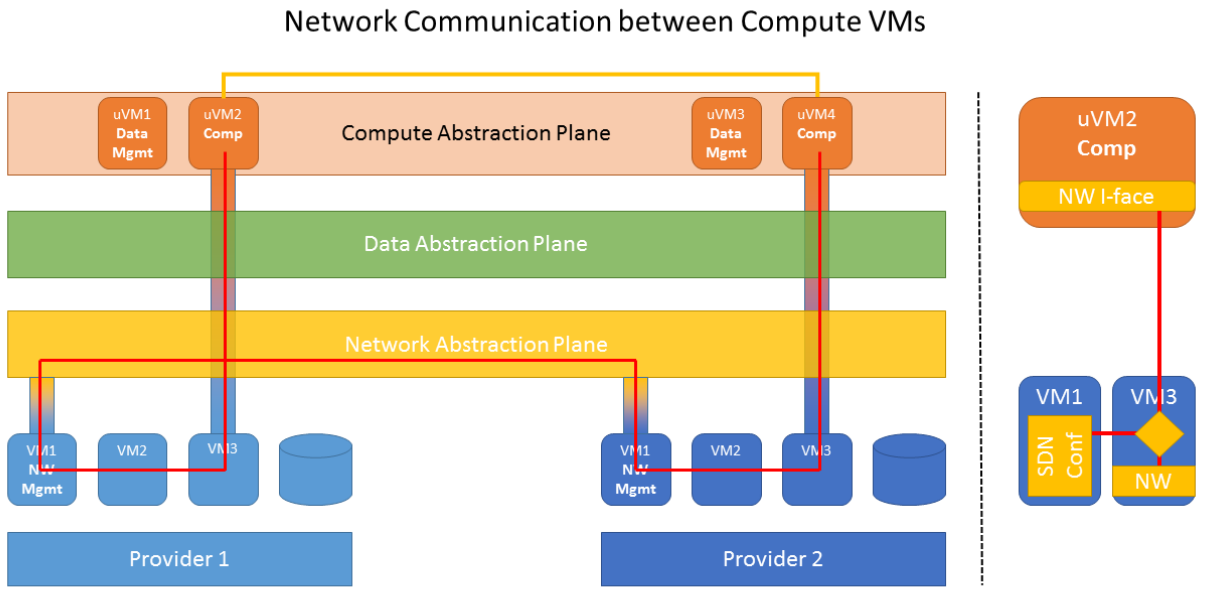

Figure 5.3: Network architecture interfacing scenario

In SUPERCLOUD, SDN provides an abstract view on the network, where the network application can perform any kind of forwarding while hiding the details of the underlying physical hardware. In order to enable the communication between two user VMs on different cloud providers, user VMs must communicate through their host VMs located on their cloud providers. In Figure 5.3 uVM2 and uVM4 are located on different providers and aim to communicate with each other. A host VM then uses the interface provided by the network management VM to forward traffic to the desired destination. The network management VM takes care of the communication between different user VMs on different platforms through SDN controllers. On the right side of Figure 5.3 the network components involved in a communication are shown in more detail. Each User-VM has a network interface which provides the view of a local network between all uVMs of one U-Cloud. In the underlying VM of each uVM a SDN is operating which provides channels between all providers used by a U-Cloud, i.e., allowing all uVMs to communicate with each other. In most instances of SDN the separation of the control logic and network infrastructure is realized over the OpenFlow protocol [43]. OpenFlow is the communication protocol that allows the controller to centrally control the behavior of the network switches.
The whole process is invisible to the user VMs on the compute plane. The network abstraction provides a logical interface for forwarding and distribution of the traffic through the network plane. User VMs communicate with each other as they all reside on the same provider.

5.2.4 Security Self-Management Interfaces

As already mentioned in previous chapters, SUPERCLOUD intends to enable users to customize their security settings even if their SUPERCLOUD deployment is spread across different CSPs based on their own protection requirements. Moreover, as a self-managed cloud it should maximize the

automated administration mechanisms without sacrificing the flexibility of the users in defining their own protection policies.

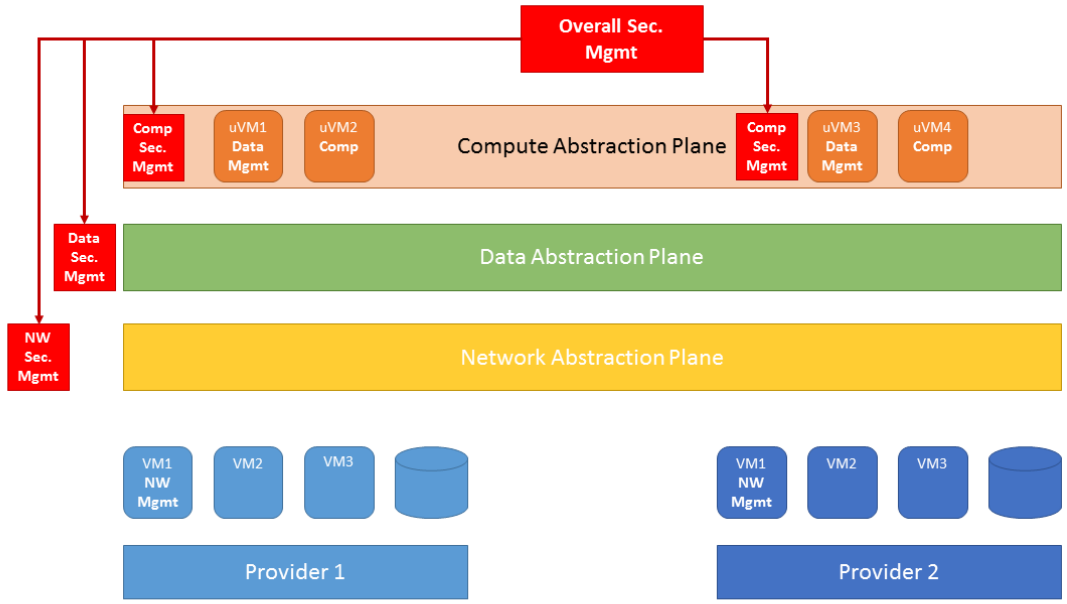

Figure 5.4: Security self-Management interfacing scenario

The Security Self-Management is distributed across the abstraction planes of SUPERCLOUD. It will allow the users to define fine-grained security parameters and functions on each plane, through the connections and integrated components in the compute, network and data planes.
As illustrated in Figure 5.4, within each plane's architecture, security components are dedicated to the execution of infrastructure monitoring and auditing, handling of the configuration and deployment policies, authorization and resource access control, and the management of security attributes. The overall security management component is charge of multi-cloud security orchestration. It also contains some specific security services such as horizontal security management across providers. As shown in Figure 5.4 a compute security management component exists per cloud service provider, in charge of vertical security management across infrastructure layers. In Figure 5.4 each plane is equipped with a security management component which is connected to the overall security management.

5.2.5 Architecture Plane Interfaces

Figure 5.5 shows the interfaces between the planes and the providers which were identified in the previous Sections 5.2.1 – 5.2.4. The interfaces are shown by the connections between the planes / VMs. The two colors of the connections represents the components an interface is connecting.
In summary, every provider has a network management VM that conducts communication through network abstraction plane among different providers, VMs and planes. The data abstraction plane provides interfaces to the network and compute abstraction planes to let them access the local and provider-specific storage resources. Last but not least, user VMs on the compute plane interact with their corresponding VMs on their providers to perform various tasks, or to communicate with other user VMs.

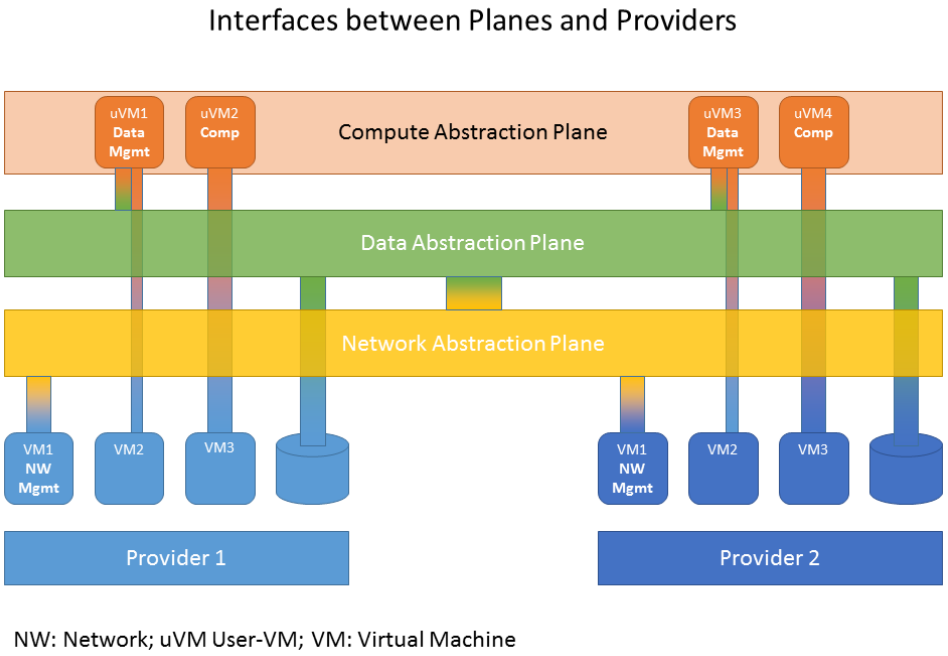

Figure 5.5: Plane / provider interfaces

5.3 Overall Architecture – Deployment Architecture

The overall SUPERCLOUD architecture is shown in Figure 5.6. For simplicity, it shows the deployment of two U-clouds, U-cloud A marked by the violet dashed line and U-cloud B in cyan. For U-Cloud A details of the deployment are shown. Again for simplicity, U-cloud A is deployed across only two Cloud Service Providers (CPS), a public cloud (Figure 5.6 on the left side; coloured light blue) and a private cloud (on the right side; coloured dark blue in Figure 5.6).
Private and public clouds both provide basic resources like virtual machines (VMs) in which computations can be performed, storage for data and network connectivity for the VMs. However, the private and public clouds differ in the extent of control they provide over these resources. The public cloud does not give direct and full access to the storage and network resources but allows their use only through Application Programming Interfaces (APIs). In contrast, a private cloud gives the cloud user more control, for instance, by giving the user access to the SDN controller so that the user can configure the network for her U-Cloud as desired. Similarly, storage services of public cloud provide access only at a higher level (e.g., at file system level) while private clouds provide more control (e.g., by providing storage as block devices). Also the control over the compute environments (VMs) differs; In public cloud the CPS retains full control over the underlying virtualization software (hypervisor) while in a private cloud the user can be provided with the ability to leverage the virtualization layer for his own goals. In Figure 5.6 the User-centric System Services (USS) are used to transparently mange the network of the VMs through Open vSwitches (OvS).
Although the resources and control capabilities provided by the different cloud service providers are utilized by SUPERCLOUD, they are at the same time abstracted from the higher layers of the architecture. In particular, the SUPERCLOUD architecture is divided into two layers. The *Lower Layer* (LL) is provided by the cloud service providers. Especially, the virtual machines executed directly on the cloud provides infrastructure are part of the Lower Layer. The *Higher Layer* (HL) of the SUPERCLOUD architecture offers abstraction and transparency towards the SUPERCLOUD-user.
The Higher Layer is most important to provide the new functionalities SUPERCLOUD offers. For compute, data and network planes, HL provides an abstraction for the underlying resources. For the

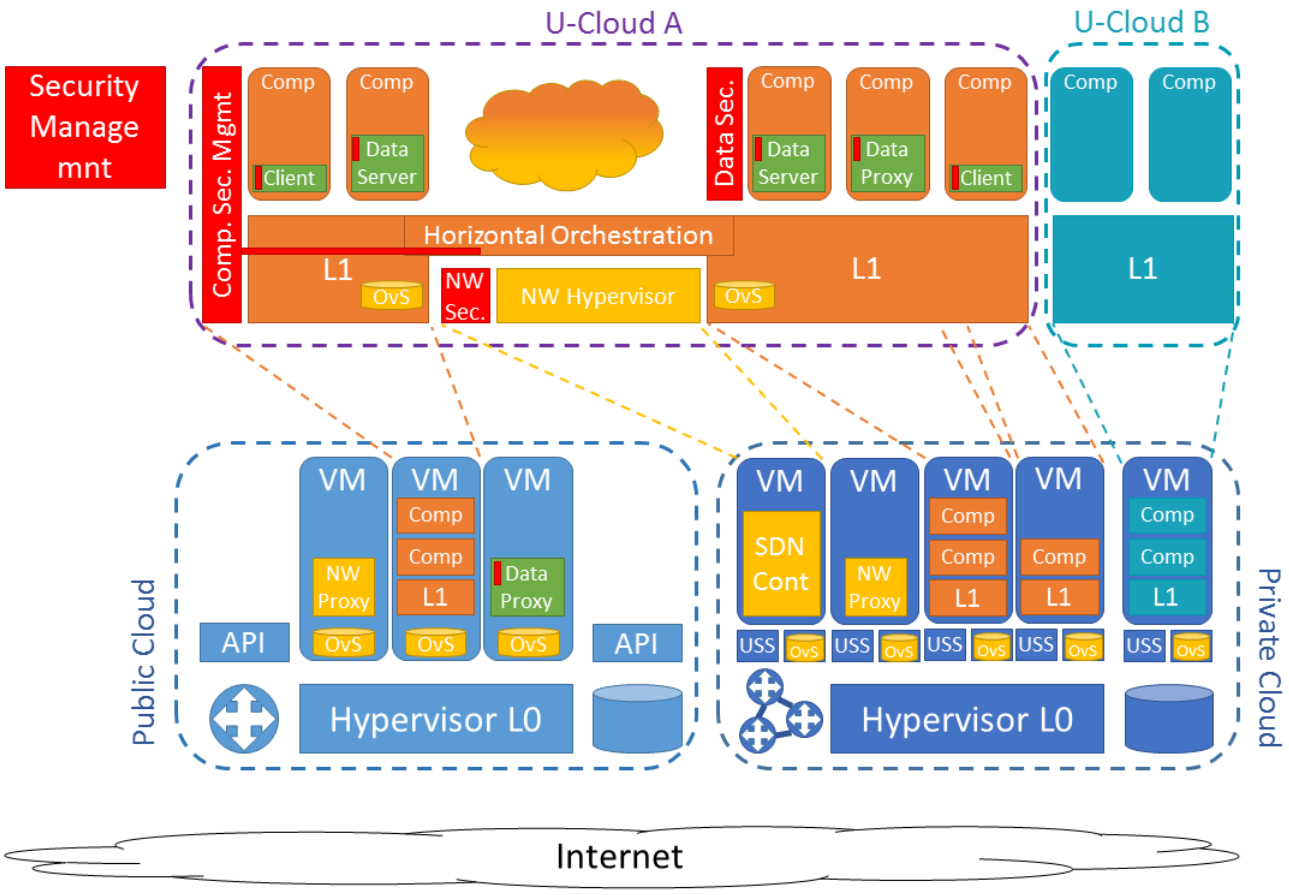

Figure 5.6: SUPERCLOUD deployment architecture

compute plane, a *nested* virtualization layer provides horizontal orchestration of the compute resources. In Figure 5.6 This additional layer is labelled L1; as shown in Figure 5.6 compute environments (or execution environments) in the U-Cloud A are run on-top of the L1 layer. Hence, the actual allocation of SUPERCLOUD-user execution environments (EE) to VMs in the Lower Layer is hidden. Each VM on LL can host one or more compute environments; each VM hosting an EE runs also the L1 virtualization layer. The details for the compute architecture are explained in deep detail in Deliverable D2.1.

Beside compute abstraction and orchestration the Higher Layer of SUPERCLOUD provides network abstractions. Figure 5.6 shows a network hypervisor (NW Hypervisor) which provides transparent connections between all components of the SUPERCLOUD HL. Each VM is therefore equipped with an Open vSwitch (OvS) component which can be configured via an SDN controller allowing to form a private network for the Higher Layer of SUPERCLOUD. To provide horizontal orchestration across different CSPs a network proxy (NW proxy in Figure 5.6) is used which connects all cloud providers a U-cloud spans across. A more detailed description of the network architecture is provided in Deliverable D4.1.

The data abstraction of SUPERCLOUD is provided by a client component in each compute environment of the Higher Layer of SUPERCLOUD along with data servers and data proxies. The data proxies can also be located in VMs in the Lower Layer. The data client provides the abstraction of the data architecture towards the SUPERCLOUD user. Data servers and proxies are responsible for the orchestration of the data plane. Details on the data abstraction plane are described in Deliverable D3.1.

The security management of the SUPERCLOUD architecture provides, similar to the compute, data and network planes, abstraction and orchestration of security functions and configurations (shown on the left in Figure 5.6). Each of the sub-architectures is equipped with a security management

component (Comp. Sec. Mgmt, NW Sec. and Data Sec in Figure 5.6). The per-plane security components interact with and monitor the security units of the corresponding functional components of the plane. For instance, in Figure 5.6 each component of the data plane (shown in green) has a dedicated embedded security component, which are organized and managed by the Data Security Management component.

Chapter 6 Architecture Mapping to Use Cases

This chapter describes on an abstract level the mapping of the architecture to two specific use cases from the healthcare domain.
This description considers two viewpoints: a technical point of view, showing which components are needed for which functionality and how the architecture satisfies all requirements from healthcare; and a user point of view, explaining how the architecture is used by someone who wants to bring health care into the cloud and how he can achieves compliance with, e.g., legal requirements.

6.1 Philips Healthcare: Medical Imaging Platform

Use cases in healthcare, e.g. Philips Healthcare system, require dealing with large quantity of data especially when it comes to imaging studies. Hospitals will benefit from a cloud data storage, which will help them to store, manage and process clinical data more efficiently. The cloud architecture should be designed so that the risks of security breaches and privacy violations are minimized. It must provide certain security measures to prevent unprivileged access to data during processing sessions, and while residing in storage, with regard to the defined policies. These policies might include hospital context, legal country boundaries and user groups. Moreover, isolation technologies should be adapted to counter any attacks that aim to tamper applications running in the cloud. In terms of performance, robust data processing with low latency is desired, especially across different clouds.
In the overall architecture, abstractions made on the compute, network and data planes will help to realize the above objectives. Data access is facilitated using data management VMs on the compute plane, as well as functionalities incorporated in the data abstraction plane. Furthermore, security is managed through dedicated security components on each plane. The provider has only a limited control over the infrastructure to avoid inspection or analysis of patient data or exfiltration of EE instances without explicit user consent. Users can decide to grant permissions to other users or the cloud provider to share information or enable specific services. Isolation technologies applied to the overall architecture will prevent any unprivileged access to the user VMs from malicious VMs or even the cloud provider itself, in a privacy-compliant manner.

6.1.1 Example: Disaster Recovery Use-Case

Hospitals can store their clinical data as well as their imaging studies in an on-premise private cloud storage. Through archiving in the cloud, it becomes easier for them to manage their patient data. These clinical data, especially imaging studies can be as large as 1GB per file. Since storage size on-premise can be limited, it makes more sense to store this data into a public cloud. In order to have more flexibility, the data from the last 6 months can be stored on the private cloud which is on premise, whilst the public cloud is used to store data for the longer period (e.g., 10+ years).
Fig. 6.1 shows an example of using SUPERCLOUD to implement the mentioned solution. Three different hospitals (A, B and C) together use a private cloud to store and manage their clinical data. A virtual machine on the compute plane is dedicated to each of the hospitals for its operations, Whenever a hospital VM wants to store or retrieve a clinical data, e.g. MRI imaging, it communicates with a *Picture Archiving and Communication System (PACS)* virtual machine that runs on the compute plane of the SUPERCLOUD and is an interface to the data plane.

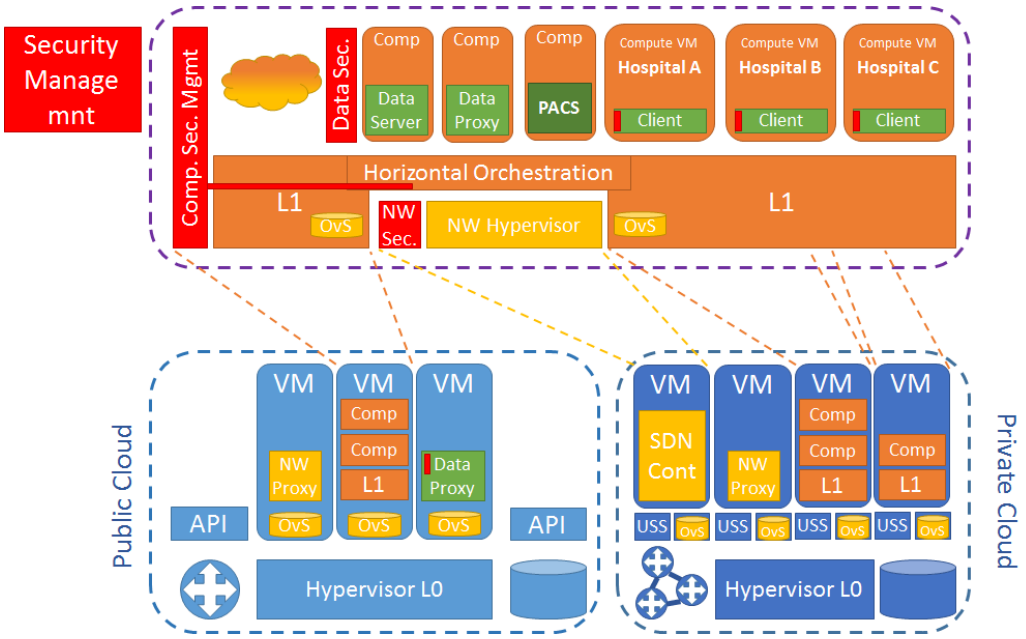

Figure 6.1: Disaster recovery use case and the overall architecture of the SUPERCLOUD

Users of the SUPERCLOUD can benefit from both private and public clouds (e.g. Amazon EC2). In this use-case, the public cloud can provide long-term storage for the hospitals. However, a user VM, or more precisely the PACS VM, only needs to communicate with the data plane, to get or store the data. The data plane provides an abstraction to the user VM, where the underlying storage is invisible to the user VM, and the data is transparently encrypted. Any data that is older than 6 months is stored on the public cloud, whereas recent data will be kept on the private cloud's on premise storage, providing instantaneous access to it. Here, the network plane is responsible for handling all the communication across different clouds and VMs.
Hospitals can also define their security policies, e.g. how their data is accessed by other hospitals. Those components that are dedicated to data security, and security management across L1 hypervisor and compute VMs will prevent any unprivileged access to data based on each hospital's defined security policies.

6.2 MaxData: Healthcare Laboratory Information System

The healthcare laboratory information system, which is called CLINIdATA®LIS, is a cross-platform web application that stores medical data along with other personal data. The users of this system range from small laboratories with a few dozens of professionals and hundreds of transactions per day, to very large hospital clusters with thousands of professionals and tens of millions of transactions per day. It is intended to integrate CLINIdATA®LIS with different types of other clinical and non-clinical information systems across different organizations, yet enabling them to control and define the way their resources and data will be protected against unprivileged access.

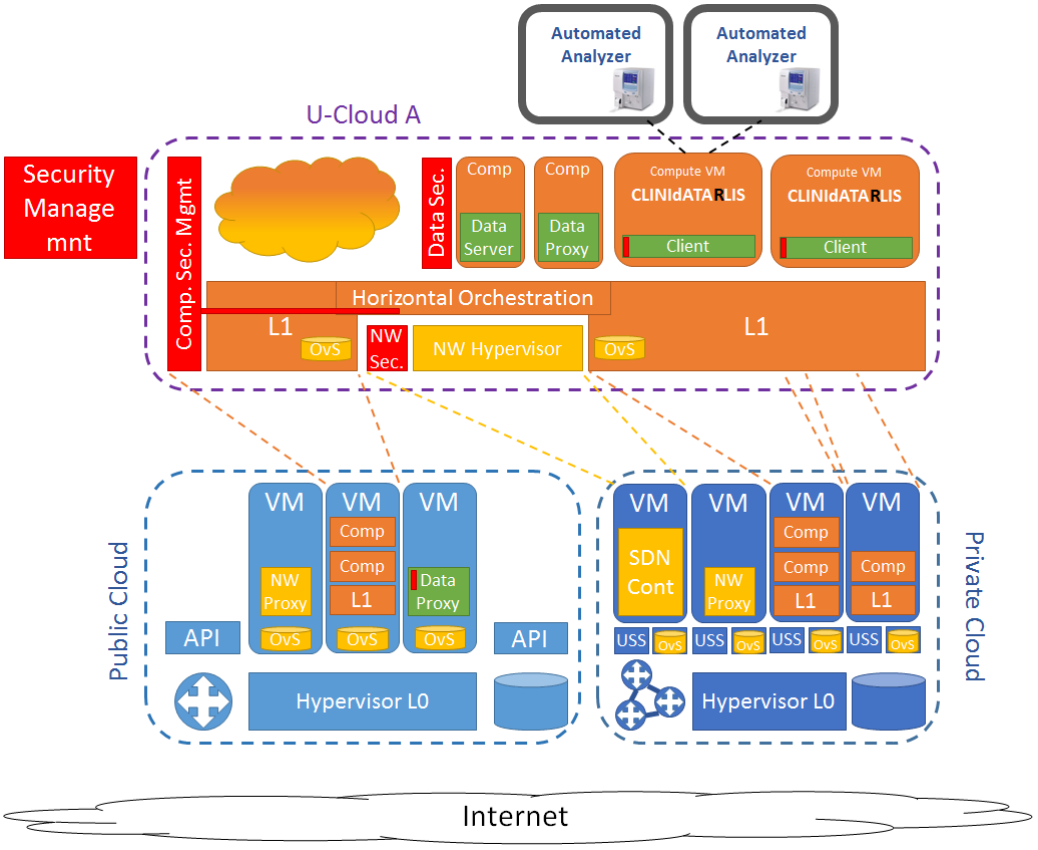

Figure 6.2: MaxData use case and the overall architecture of the SUPERCLOUD

Since CLINIdATA®LIS is used to obtain highly valuable and life-saving information about patients and clinical procedures, it needs to be constantly available to ensure operation of various departments including the ER (emergency room). As a result, the SUPERCLOUD infrastructure must provide high availability whenever necessary, e.g. through data replication, in accordance to the data-owner's configured security policies.
Figure 6.2 shows a high level adaptation of the SUPERCLOUD's architecture to this use case. CLINIdATA®LIS dedicated EEs provide real-time interfaces to physical electronic healthcare equipments, namely automated analyzers. The SUPERCLOUD architecture enables users to configure through dedicated security components integrated into each abstraction plane the security policies used to control access to their data. On the other hand, the seamless integration of different providers and data owners under one architecture will provide concrete availability of data for requesting entities. The following describes how the MaxData requirements are mapped to the SUPERCLOUD's requirements:

- **High Availability:** The architecture provides appropriate facilities to enforce availability requirements regarding the redundancy, backups, and disaster recovery measures for data resources as specified in SLA between the user and CSP. It will be provided through the data abstraction plane.

- **Real-Time:** SUPERCLOUD must guarantee performance for on-line user interaction or for interactions with external systems, e.g. physical electronic equipment. It includes providing facilities in the compute, data and network planes to guarantee predictable and bounded response times for time-critical computations, data communication as well as storage access times.

- **Control over Architecture and Data:** The architecture must enable the user to actively monitor its allocated data assets and computational resources, while providing a high level of

customizability of the storage architecture and its services. This will be done through dedicated security components in the data and compute planes.

- **Location-awareness:** In order to satisfy legal requirements that may prohibit the transfer of data across jurisdictional boundaries, SUPERCLOUD data plane must provide the means for policing the physical location of the data storage, as well as enabling users to monitor their allocated resources with regard to their physical location.
- **Location-control:** Users must have the possibility to define the set of possible physical locations, at country level, where users' data may be stored or though which the data may transit. This must be done in line with the security settings defined or controlled by the user. The data and network planes must provide the necessary facilities for enforcing and monitoring the user-provided settings.

6.3 Use Case Requirements Mapping

Table 6.1 shows the high level mapping between SUPERCLOUD requirements to the corresponding use cases and the abstraction planes that are responsible of implementing facilities for responding to these requirements. As the design of the individual sub-architectures is refined, this mapping will be updated to reflect the actual sub-architecture components responsible realizing the requirement.

Table 6.1: Mapping of use case requirements and SUPERCLOUD abstraction planes

Requirements		Use Cases	Realized by
Functional	Provider and Platform Independence		compute, data and network planes
	Isolation between Users	PHHC	
Reliability	Integrity and Completeness of Data	PHHC	data plane
	Availability	MaxData	compute, data, and network planes
	Performance	PHHC MaxData	
Security	User-Controlled Security Settings	PHHC MaxData	compute, data, and network planes
	Location-Aware Control	PHHC MaxData	data and network planes
	Privacy of User Data	PHHC	compute, data, and network planes
	Accountability	PHHC MaxData	
Manageability	Interoperability	PHHC	compute, data, and network planes
	Legacy Support		

Chapter 7 Summary and Conclusion

We have presented the preliminary high level architecture specification of SUPERCLOUD that will be refined and developed as the work in the SUPERCLOUD project proceeds and new insight concerning the practical implementation of these components becomes available. The preliminary architecture consists of three abstraction planes: the compute plane, the data plane and the networking plane. The purpose of each of these planes is to provide abstractions of computation, data storage and data communications services used by execution environments that SUPERCLOUD users can run on the SUPERCLOUD infrastructure. The ensemble of such services, a so-called U-Cloud can be distributed over the cloud services of several different cloud service providers (CSPs). By combining services of private CSPs potentially providing a high degree of customizability with the offerings of public CSPs providing typically high-capacity and low-cost services, U-Clouds with desirable properties for individual users can be instantiated.

The interfaces between the sub-architectures realizing the abstraction planes are defined to be narrow to provide stability while allowing each sub-architecture to be developed independently in parallel. In the next steps of the project, the sub-architectures are being refined and implemented in the respective work packages WP2, WP3 and WP4 in close co-operation with the activities of work package WP5, in which the use cases are being developed.

Another major activity related to the SUPERCLOUD architecture is the expansion of the security self-management architecture along the lines already set out in Deliverable D1.2. The security management framework connects to each sub-architecture and will provide the facilities for allowing each SUPERCLOUD user to adapt fine-grained security settings for data, computation and communications inside her own U-Cloud.

Chapter 8 List of Abbreviations

API	Application Programming Interface
CaaS	Crypto-as-a-Service
CPU	Central Processing Unit
CSP	Cloud Service Provider
DA	Direct Accessor
DCC	Distributed Cloud Computing
DCI	Distributed Cloud Infrastructure
EC	European Commission
EE	Execution Environment
ER	Emergency Room
GRE	Generic Routing Encapsulation
IaaS	Infrastructure-as-a-Service
IETF	Internet Engineering Task Force
IP	Internet Protocol
LLDP	Link Layer Discovery Protocol
LO	Layer Orchestrator
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
NAS	Network Attached Storage
NIC	Network Interface Card
NVGRE	Network Virtualization using GRE
NVP	Network Virtualization Platform
OvS	Open vSwitch
OVSDB	Open vSwitch Database Management Protocol
OVX	OpenVirteX
PaaS	Platform-as-a-Service
PHHC	Philips Healthcare
RBridges	Routing Bridges
SDN	Software-Defined Networking
SLA	Service Layer Agreement
SMM	System Management Mode

SSC	Self-Service Cloud Computing
STT	Stateless Transport Tunneling
SVM	Secure Virtual Machine
TBD	to be determined
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TRILL	Transparent Interconnection of Lots of Links
USS	User-centric System Services
VLAN	Virtual LAN (Local Area Network)
VPC	Virtual Private Cloud
VPN	Virtual Private Network
VM	Virtual Machine
VXLAN	Virtual eXtensible Local Area Networks

Bibliography

[1] Alephcloud. https://www.crunchbase.com/organization/alephcloud-systems#/entity/.

[2] Boxcryptor. https://www.boxcryptor.com/.

[3] CipherCloud. https://www.ciphercloud.com/.

[4] Data Masker. https://www.datamasker.com/.

[5] Floodlight. http://www.projectfloodlight.org/floodlight/. Accessed on 23.11.2015.

[6] IMB Optim. https://www.ibm.com/software/data/optim/.

[7] Informatica Data Privacy. https://www.informatica.com/products/data-security/data-masking.html.

[8] MEGA. http://www.mega.com/.

[9] MultCloud. https://www.multcloud.com/.

[10] Oracle Data Masking Pack. http://www.oracle.com/us/products/database/data-masking-subsetting/overview/index.html.

[11] Otixo. https://www.otixo.com/.

[12] Perspecsys. https://perspecsys.com/.

[13] IBM IDentity Mixer, 2015.

[14] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow. OpenVirteX: Make your virtual SDNs programmable. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, 2014.

[15] Amazon Web Services. AWS Config. Available at https://aws.amazon.com/config/, 2014.

[16] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky. Hypersentry: Enabling stealthy in-context measurement of hypervisor integrity. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 38–49, New York, NY, USA, 2010. ACM.

[17] T. Benson, A. Akella, and D. Maltz. Unraveling the complexity of network management. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 335–348, Berkeley, CA, USA, 2009.

[18] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. Tvdc: Managing security in the trusted virtual datacenter. *SIGOPS Oper. Syst. Rev.*, 42(1):40–47, Jan. 2008.

[19] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa. DepSky: Dependable and secure storage in cloud-of-clouds. *ACM Transactions on Storage*, 9(4), 2013.

[20] A. Bessani, R. Mendes, T. Oliveira, N. Neves, M. Correia, M. Pasin, and P. Verissimo. SCFS: a shared cloud-backed file system. In *Proc. of the 2014 USENIX ATC*, 2014.

[21] A. N. Bessani, J. Sousa, and E. A. P. Alchieri. State machine replication for the masses with BFT-SMART. In *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014*, pages 355–362, 2014.

[22] S. Bleikertz, S. Bugiel, H. Ideler, S. Nürnberger, and A.-R. Sadeghi. Client-controlled cryptography-as-a-service in the cloud. In *Applied Cryptography and Network Security*, pages 19–36. Springer, 2013.

[23] S. Bleikertz, T. Groß, and C. Vogel. Cloud Radar: Near Real-Time Detection of Security Failures in Dynamic Virtualized Infrastructures. In *Annual Computer Security Applications Conference (ACSAC 2014)*. ACM, Dec 2014.

[24] S. Bradner. Rfc2119: Key words for use in rfcs to indicate requirement levels.

[25] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, and T. Schneider. Twin clouds: Secure cloud computing with low latency. In *Communications and Multimedia Security Conference (CMS'11)*. Springer, 2011.

[26] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. Self-service Cloud Computing. In *ACM Conference on Computer and Communications Security*, CCS'12, pages 253–264, New York, NY, USA, 2012. ACM.

[27] M. Casado. OpenStack and Network Virtualization, April 2013.

[28] B. Davie and J. Gross. A Stateless Transport Tunneling Protocol for Network Virtualization (STT). Internet Draft draft-davie-stt-06 (Informational), Apr. 2014.

[29] D. Dobre, P. Viotti, and M. Vukolic. Hybris: Robust hybrid cloud storage. *SoCC*, 2014.

[30] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. *Internet Computing, IEEE*, 17(2):20–27, 2013.

[31] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784 (Standards Track), Mar. 2000.

[32] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Proc. Network and Distributed Systems Security Symposium (NDSS)*, February 2003.

[33] P. Garg and Y. Wang. NVGRE: Network Virtualization Using Generic Routing Encapsulation. RFC 7637 (Informational), Sept. 2015.

[34] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[35] J. Gross, T. Sridhar, P. Garg, C. Wright, I. Ganga, P. Agarwal, K. Duda, D. Dutt, and J. Hudson. Geneve: Generic Network Virtualization Encapsulation. Internet Draft draft-ietf-nvo3-geneve-00 (Informational), May 2015.

[36] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. *Comp. Comm. Rev.*, 2008.

[37] A. Herzberg, H. Shulman, J. Ullrich, and E. Weippl. Cloudoscopy: Services Discovery and Topology Mapping. In *ACM Workshop on Cloud Computing Security*, CCSW'13, pages 113–122, New York, NY, USA, 2013. ACM.

[38] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang. Network virtualization in multi-tenant datacenters. In *11th USENIX Symposium on Networked Systems Design and Implementation*, NSDI '14, 2014.

[39] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[40] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: a comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, January 2015.

[41] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348 (Informational), Aug. 2014.

[42] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 10:1–10:1, New York, NY, USA, 2013. ACM.

[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38(2):69–74, Mar. 2008.

[44] MIKELANGELO H2020 Project. `http://www.mikelangelo-project.eu/`.

[45] A. Nguyen, H. Raj, S. Rayanchu, S. Saroiu, and A. Wolman. Delusional Boot: Securing Hypervisors Without Massive Re-engineering. EuroSys'12.

[46] OpenStack Watcher. `https://wiki.openstack.org/wiki/Watcher/`.

[47] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 238–252, Washington, DC, USA, 2013. IEEE Computer Society.

[48] R. Perlman. Rbridges: transparent routing. In *Proceedings of IEEE INFOCOM*, INFOCOM '04, 2004.

[49] R. Perlman, D. Eastlake, D. Dutt, S. Gai, and A. Ghanwani. Routing Bridges (RBridges): Base Protocol Specification. RFC 6325 (Standards Track), July 2011.

[50] B. Pfaff and B. Davie. The Open vSwitch Database Management Protocol. RFC 7047 (Informational), Dec. 2013.

[51] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The Design and Implementation of Open vSwitch. In *12th USENIX Symposium on Networked Systems Design and Implementation*, NSDI '15, 2015.

[52] L. Popa, M. Yu, S. Y. Ko, S. Ratnasamy, and I. Stoica. Cloudpolice: Taking access control out of the network. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 7:1–7:6, New York, NY, USA, 2010. ACM.

[53] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. V. Doorn, J. L. Griffin, S. Berger, R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. Doorn, J. Linwood, and G. S. Berger. sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. In *IBM Research Report RC23511*, 2005.

[54] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu. Policy-sealed Data: A New Abstraction for Building Trusted Cloud Services. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 10–10, Berkeley, CA, USA, 2012. USENIX Association.

[55] J. Schiffman, Y. Sun, H. Vijayakumar, and T. Jaeger. Cloud Verifier: Verifiable Auditing Service for IaaS Clouds. In *Proceedings of the IEEE 1st International Workshop on Cloud Security Auditing (CSA 2013)*, June 2013.

[56] S. Shenker. Stanford Seminar - Software-Defined Networking at the Crossroads, June 2013.

[57] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010.

[58] U. Steinberg and B. Kauer. NOVA: A Microhypervisor-based Secure Virtualization Architecture. In *5th European Conference on Computer Systems*, EuroSys'10, pages 209–222, New York, NY, USA, 2010. ACM.

[59] D. Williams, H. Jamjoom, and H. Weatherspoon. The Xen-Blanket: virtualize once, run everywhere. In *Proceedings of the 7th ACM European Conference on Computer Systems*, pages 113–126. ACM, 2012.

[60] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 203–216, New York, NY, USA, 2011. ACM.