A recursive orchestration and control framework for large-scale, federated SDN experiments: the FELIX architecture and use cases

Programmable networks are a substantial part of current R&D on Future Internet (FI) in Europe and worldwide, with considerable impact generated by large-scale testbed infrastructures. In such testbeds, researchers validate proof-of-concept prototypes for new algorithms and mechanisms for efficiently controlling and managing network resources. One of the key domains for FI research is Software-Defined Networking (SDN), which creates innovations in existing Internet architectures by shifting the control and logic outside the network equipment to Data Centres. International cooperation among leading research centres in Europe, Americas and Asia is key to validate SDN foundations and tools. EU and Japan have jointly funded the FELIX project (FEderated Test-beds for Large-scale Infrastructure eXperiments), which defines a common control and orchestration framework to manage federated FI testbeds across continents. This framework enables an experimenter to i) request and obtain resources across different testbed infrastructures dynamically; ii) manage and control the network paths connecting the federated SDN testbeds; iii) monitor the underlying resources; and iv) use distributed applications executed on the federated infrastructures. This paper describes the high-level architecture of the FELIX framework and details six use cases that will be employed for validation. We present our analysis and end-user considerations, highlighting the necessity for resource accessibility and coherent use of physical connections over a large-scale testbed where different control technologies like OpenFlow and the Network Service Interface (NSI) are simultaneously used.

Keywords: Software-Defined Networking; federated infrastructures; large-scale testbeds; Network Service Interface; OpenFlow

1 Introduction

Programmable networks, based on Software-Defined Networking (SDN) principles, decouple the control and data planes and allow for remote software to assume the control and management of the underlying network. These networks are a substantial part of existing R&D on the Future Internet (FI) in Europe and worldwide. Both academic and industrial SDN researchers around the world are embracing large-scale testbed infrastructures to validate their proof-of-concept prototypes and experiment with new algorithms, protocols or network functions in large-scale, efficient, predictable, realistic environments.

In general, FI testbeds differ in the resources provided as well as in the geographic regions in which they operate. With the aim of promoting the use of heterogeneous resources across different infrastructures, FI testbeds usually provide the experimenter with common interfaces and workflows. This is typically referred to as *testbed federation* and is a way of abstracting different internal infrastructures, resources, and procedures to enable the definition of larger experiments with unified resources that are handled the same way. An open federation of heterogeneous testbeds is non-trivial, however, and requires the design of a suitable architecture and framework. The FELIX [1] project aims to facilitate the federation and integration of different network and computing resources residing in a multi-domain heterogeneous environment across different continents. To achieve this, the FELIX architecture extends and advances assets previously developed in other FI projects (e.g. OFELIA), for instance by realizing the federation concepts defined in SFA [2] and implemented by GENI [3]. In particular, FELIX uses a combination of recursive and hierarchical configurations for orchestration, request delegation and inter-domain dependency management. Resource orchestrating entities are responsible for the synchronization of resources available in particular administrative domains. These entities and other key building blocks of the FELIX architecture are introduced in the following sections.

This paper details six use case scenarios for validating and demonstrating the FELIX framework over its distributed SDN virtual infrastructure spanning multiple domains. These use cases are grouped in two major domains: *Data Domain* and *Infrastructure Domain*. The Data Domain use cases focus on the efficient use of SDN technologies to provide interconnections across geographically dispersed testbeds with the ability to realize data migration dynamically and efficiently. The Infrastructure Domain use cases are mainly oriented towards the use of a virtual distributed infrastructure that can be employed for migrating entire data processing workloads. This paper reports early-phase work focusing on use-case identification [4] and architecture definition [5]. Future work will address the validation of these use cases on the FELIX federated infrastructure.

The remainder of this paper is organized as follows: Section 2 describes the different resources and key concepts considered in the FELIX experimental facility, Section 3 is an in-depth discussion into the FELIX architecture, Section 4 details the use cases considered, and Section 5 presents conclusions and future work.

2 Resources in the FELIX Federated Testbed

Resources in FELIX include both networking and computing capacities available at geographically dispersed facilities. Resources are under the administrative control of different but cooperating (federated) stakeholders. Federated resources in FELIX are used to create a virtual infrastructure that spans multiple domains. Note that this environment is starkly different from the case of a) a single administrative domain with resources geographically distributed across the world, e.g. data centres of a single cloud operator; and b) loosely coupled, interconnected islands that allow for remote access to certain resources. FELIX is primarily interested on network enablers and, in particular, integration of SDN testbeds with Network Service Interface (NSI) [6] controlled transit domains, with particular focus on the Connection Services (NSI-CS). These services, in turn, can be used to solve the dynamic establishment and teardown of network connectivity services (based on L2 switching and L3+ flow routing/forwarding) across multiple domains and technologies.

2.1 Virtual Infrastructures through Federation

Monga et al. [7] note that connecting facilities at continental and inter-continental scale is not a trivial task and they motivate the need for connecting facilities (such as those considered in FELIX) at the lower layers (e.g. L2), thus avoiding the system overheads introduced by the connections established at L3 and above. However, the resulting proposals [7], [8], [9] do not conform to emerging standards, such as NSI. Moreover, previous work does not comprehensively consider the elements of each island from a network control perspective, and does not account for policies and trust. We believe that these aspects will play a crucial role in determining the adoption of a framework suitable for federated resources. Our analysis of the latest research literature on this topic has highlighted the need to introduce new APIs and logic for globally distributed heterogeneous facilities (e.g. OFELIA islands and JGN-X RISE testbeds). It is clear that these new APIs and logic should capitalize on SDN and NSI mechanisms and protocols to facilitate the dynamic, on-demand establishment of end-to-end, cross-continental virtual network infrastructures.

While SDN testbed infrastructures are constructed from the viewpoint of network research and development, computing and storage resources are also important components in each testbed. FI services can be grouped into two categories: those that use network resources to move data, and those that use the whole infrastructure (including computing and storage resources) to provide network-based services. Therefore, we consider two major classes of use cases for the demonstration of virtual infrastructure based on federated testbed resources. Namely, the first category of use cases belongs to the *data domain* since the primary focus here is the use of data. The second category of use cases forms part of the *infrastructure domain*, which includes the three resource types in a testbed: networking, storage, and computing.

2.2 Key System Concepts and Definitions

The foundation of the FELIX experimental facility consists of the key system concepts summarized in this subsection.

FI experimental facilities (or SDN-controlled network domains) are controlled by dedicated software, exposing interfaces that can be used by a federation framework to orchestrate resources in a multi-domain environment. The SDN-controlled network domains are illustrated in Figure 1.

An *SDN island* is a set of virtualized network and computing resources under the same administrative ownership and control. It may consist of multiple SDN zones, each characterized by a specific set of control tools and interfaces. Each *SDN Zone* is a set of resources grouped together by common technologies and/or control tools and/or interfaces, e.g. L2 switching zone, optical switching zone, OpenFlow protocol controlled zone, and other transit domain zones with a control interface. The major goal of defining SDN zones is to implement appropriate policies for increasing availability, scalability and control of the different resources of the SDN islands. Examples of zone definitions can be found in widely deployed Cloud Management Systems (CMS) such as CloudStack, where infrastructure is partitioned into regions, zones [10], pods, and so on. In addition, OpenStack offers infrastructure partitioning through availability zones and host aggregates [11].

Transit network domains use NSI to expose either automatically or on-demand control of the connectivity services and, optionally, exchange inter-domain topology information. On-demand interconnectivity with a specific granularity must be provided in order to federate resources that belong to distant experimental facilities. In FELIX, it

is assumed that all experimental facilities will be interconnected with networks running NSI-compatible network controllers. The NSIv2.0 standard interface will be used as a means to orchestrate network resources for an experiment setup.

In Figure 1, a *slice* is a user-defined subset of virtual networking and computing resources. Each slice is an abstraction created upon the set of physical resources available in the federated SDN Zones and SDN Islands. Every slice is isolated from other slices running simultaneously on the same physical resources, thereby avoiding interferences from other separate experiments. A slice should also be dynamically extensible across multiple SDN Islands. Each slice instantiates only when needed the specific set of control tools required for the specific zones it must traverse.

The slice concept originates from the Slice-Based Federation Architecture (SFA) that defines a number of abstractions to identify provisioned resources, enable their aggregation and identify entities to manage resources (i.e. slivers, slices and component managers). SFA also provides a minimal set of structures and interfaces, which consist of two parts: 1) a specific data type per resource encapsulated in a RSpec to define, for example, a computing node, and 2) a list of methods following a specific workflow in order to reserve and provision any resource. These interfaces and data models facilitate and standardize the process of providing a federated slice composed of heterogeneous resources located in any other testbed in the federation. In order to do this, every SFA-compliant testbed must share the same interfaces and data models with its federated members to be able to understand resource requests. More information and samples can be found on the FELIX project implementation deliverables, available at [1].

3 The FELIX Architecture

The FELIX architecture is the result of a careful analysis of the state of the art in relevant FI research projects in both Europe and Japan. From the European side, the OFELIA [12], FIBRE [13] and Fed4FIRE [14] projects were taken into account thereby providing a working approach to large-scale distributed systems and federation (e.g. through SFA and GENI), as well as addressing federation between heterogeneous testbeds. From the Japanese testbeds, GridARS [15] and RISE [16] were considered in order to manage seamlessly the establishment of dynamic inter-domain communication through the NSI protocol. Taken together, we have defined a modular and multi-layer architecture for the FELIX control framework. As illustrated in Figure 2, we use the combination of two different ‘spaces’, namely the *FELIX Space* and *User Space*, which cooperate to build, manage, control, and monitor a large-scale virtual infrastructure.

3.1 Spaces in the architecture

The FELIX Space is composed of management and control tools that coordinate the creation of a virtual environment in heterogeneous, multi-domain, and geographically distributed facilities. The elements in this layer operate in both hierarchical and recursive models for efficient multi-domain information management and sharing. The User Space is composed of any tool or application a user wants to deploy to control his or her virtual network environment or to run a particular experiment within it. These two logical spaces glue together different functional building blocks, as shown in Figure 2.

In the FELIX Space, the *Resource Orchestrators* (ROs) are responsible for orchestrating the end-to-end network service and resources reservation in the entire infrastructure, as well as delegating end-to-end resource and service provisioning in a

technology-agnostic way. ROs are connected to the different types of *Resource Managers* (RMs), which control and manage different kinds of technological resources similar to the concept of Component Manager in SFA. For example, the *Transit Network RM* and *Stitching Entity RM* provide connectivity between L1/L2 transport network domains and manage physical devices by using frame, packet, or circuit switching technologies; whilst able to support different protocols. The *SDN RM* manages the user traffic environment and the network infrastructure, composed of SDN-enabled devices, by updating the flow tables of the physical devices. In addition, the *Computing RM* is responsible for setting up and configuring computing resources, i.e. creating new virtual machine instances, network interface card configuration, etc. Moreover, the FELIX Space can provide essential functionalities to the FELIX architecture using dedicated modules such as the Authentication, Authorization and Accounting (AAA) for authenticating and authorizing users, or the Monitoring Functions module to retrieve, aggregate and store metering information from networking and computing resources to be used as feedback in the experiments.

In the User Space, the *Slice Controller* can dynamically control the physical and virtual resources belonging to the user's slice environment. In other words, it can request more bandwidth, virtual CPU or RAM, add new resources such as storage, or even completely reconfigure the slice behaviour.

3.2 Common design considerations

The modules of the FELIX architecture address a number of common design requirements that are inherent and key to any successful federated Future Internet testbed.

Resource Orchestration: This entity performs the necessary orchestration of the different types of virtualized resources present in the testbeds, such as compute, storage or network nodes.

Resource Allocation Planning: This entity controls the reservation or allocation status for any resource subject to it. Both experimenter and administrator aspects are considered, for instance, allocation time desired by the experimenter or load balancing and complete provisioning cost for a given resource.

Provisioning: The system must be able to instantiate and initialize the resources requested by the experimenter. The infrastructure at the different islands must provide applications with a virtual flat environment that behaves like a dedicated cluster and in which some specific user-space resource information, like IP addresses, are made available to the experimenter after proper instantiation and configuration tasks.

Domain Resource Management: The heterogeneous resources that are provided by their corresponding resource management systems need to be managed accordingly, within that same domain.

Authentication, Authorization and Accounting: In order to achieve a controlled environment where any action is authorized and can be traced back in detail, we need to ensure that a) an actor has a valid claim on the presented identity, b) any action is exclusively performed by an authorized actor and c) every action is tracked by the accounting system.

Monitoring: The framework must provide the user with a coordinated set of monitoring data for both virtual and physical resources provisioned or located in different domains. This monitoring data is used primarily for the accounting of the infrastructure use, but also for enhanced control applications based on traffic measurements.

User Access: The federated experimental facility provides friendly interfaces to simplify the operations of the experiment lifecycle for the user, as well as facilitating the general management of the testbed resources for the administrator.

3.3 Building blocks in the FELIX Architecture

The FELIX Architecture is composed of a number of modules that implement the functionalities identified in the common design considerations. These modules or 'building blocks' are intended to be as generic as possible in order to deal with different environments. Figure 3 shows a schema with the different modules and the interactions between them.

3.3.1 Resource Orchestrator (RO)

The FELIX RO is a key element of the FELIX architecture and the cornerstone of the management and orchestration system design.

We consider that the RO operates over a federated testbed infrastructure of SDN 'islands', which are interconnected by Transit Networks subject to dynamic configuration through the Network Services Interface (NSI). The RO module is responsible for orchestrating the end-to-end network service as well as for instructing the resource reservation and provisioning for the entire FELIX infrastructure.

There are two different levels at which the RO may operate: a) the upper layer, right below the User Access level, and b) the layer immediately underneath. In a typical scenario, the RO in the upper layer can operate at continental level, whilst the ROs in the lower layer may communicate with the RMs or with other ROs.

The main functionalities of RO are as follows: i) proxying requests between the experimenter and the RMs, ii) recursively delegating requests between ROs of other federated infrastructures according to pre-defined policies, iii) maintaining an updated and aggregated topological view of its managed, underlying infrastructures, and iv) verifying proper workflow and notifying the experimenter of any detected error condition.

Request forwarding for allocating and provisioning resources is performed in a technology-agnostic way within the infrastructure and depends on the conditions defined in the federation policy engine previously configured for the particular domain. It is therefore necessary to ensure similar interfaces for each orchestrator. RO must also interpret the experimenter's request to be able to perform any request forwarding as well as evaluate the set of actions received from the user for correctness and notify the user of error conditions.

An internal view of the cross-island topology (for computing, SDN and NSI nodes) is necessary for the RO to properly forward requests, for instance by detecting where computing resources can be provisioned to meet the experimenter's requirements. Such topological information is filled from the lower layers (its scope is the resource management) to the upper layers, where this abstraction and mapping takes place. This resource discovery procedure minimizes the data being transmitted by taking advantage of the data interchanged between the modules during the expected workflow for SFA testbeds.

3.3.2 Transit Network Resource Manager (TN RM)

The TN RM enhances the FELIX architecture with mechanisms for network connectivity within and between particular domains. In order to deliver the network services in the FELIX architecture, the TN RM must be integrated with its southbound interfaces within a particular network domain. Such a domain can use different L1/L2 technologies and may be controlled by specific interfaces, systems such as the Network Management System (NMS), or protocols that are technology-dependent and unique in each case.

A single TN RM must communicate with a single RO in order to i) advertise resources under its control, ii) receive requests, and iii) notify the RO about success and failure events. A single TN RM is responsible for a group of particular network resources, which belong to a network domain and are usually managed by a single entity, i.e. a network administrator or NMS.

TN RM usually manages L1/L2 transport networks that are composed of physical devices using frames/packets or circuit switching technologies and support different protocols, e.g. MPLS/GMPLS. In order to support inter-island connectivity between existing OFELIA islands in Europe realized with VPN services over the Internet, the TN RM also supports the management of VPN set up and tear down procedures.

In the FELIX architecture, the TN RM southbound interface is based on the NSI-CS protocol for L1/L2 transport networks, a proprietary interface for L3 VPN services, whilst the northbound interface uses SFA-based APIs that can be understood by the RO.

3.3.3 Stitching Entity Resource Manager (SE RM)

The SE RM is a software element of the FELIX architecture that controls the Stitching Entity (SE), a network element providing the necessary translation mechanisms for a slice setup on top of the L2 protocol stack. SE RM hides from a user the actual complexity of the multi-domain transport network topology. SE must provide at least one of the following network functions: i) QinQ, to encapsulate slice traffic into transport network Ethernet frames, or ii) a VLAN translation mechanism to hide from users the actual VLAN tagging, which is used by carrier networks while interconnecting two or more FELIX islands.

The SE RM communicates with a single RO to i) advertise an internal topology and capabilities of the SE under its control, ii) receive requests, and iii) notify the RO about success and failure events.

A single SE RM must be implemented in each FELIX island and is responsible for single or multiple SEs, which belong to a network domain and act as an entry point to the island infrastructure.

3.3.4 SDN Resource Manager (SDN RM)

The domains in FELIX provide SDN-enabled infrastructures, usually equipped with OpenFlow-enabled network devices. This provides experimenters with tools to control their network behaviour in a programmatic way, that is, by defining a set of flow rules through a software controller that communicates with the physical network devices. Each rule defines a matching condition (any OpenFlow header, such as VLAN, and a

value to match against) as well as an action; and can be either polled from the controller by the network device, or directly inserted into special tables within the latter.

These switches and routers are configured and controlled through the SDN RM. This module can interact with a special purpose controller (e.g. FlowVisor) that is able to proxy the OpenFlow packets to the corresponding user controller thus keeping each environment for experimentation properly isolated from others. The SDN RM also allows the administrator to observe the experimenter's set of rules (which define her 'virtual network') and grant or revoke it.

3.3.5 Computing Resource Manager (C RM)

The C RM provides experimenters with mechanisms to configure, instantiate, and operate on computing resources and gives administrators the means to manage and monitor them.

This module interacts with the underlying infrastructure (virtualization servers and associated hypervisors) through an agent that acts as a proxy between the FELIX Control Framework side and the hypervisor. Therefore, the agent is able to communicate with the hypervisor and perform the operations provided by the former, such as creating, deleting and changing the status of the machines; informing of the status of each operation; and synchronizing the status of such resources between the infrastructure and management layers.

3.3.6 Authentication, Authorization and Accounting (AAA)

AAA makes available the necessary mechanisms to authenticate and authorize users and provides accounting for the user actions. These actions may be used from any other module (see Figure 3).

The FELIX architecture implements a *Certificate-Based Clearinghouse* (CH) [17] that establishes the root of a trust chain. By using a certificate-based approach, the architecture has the flexibility to federate different SDN islands easily and allows verifying the identity and privileges of all actors in the FELIX architecture.

The CH comprises a set of related services supporting AAA operations and acts as a location to lookup information about members, slices and other available services in the testbed. These services are offered through the *Member Authority* for certificate and credentials management, the *Slice Authority* for slice registry and privilege-mapping against its members, the *Project Service* for experiment registry and role evaluation, the *Logging Service* for accounting purposes and finally the *Service Registry* to register the aforementioned services. These authorities – derive from SFA principals – and the services complement the processes of i) registration and management, ii) authentication and authorization, and iii) accounting.

For federating purposes, these services can be accessed via the Common Federation API [18], allowing thus compatibility with tools like OMNI and testbeds complying with the GENI Aggregate Manager API [18].

The AAA module is closely related to the User Access and is ultimately responsible for granting access to resources, as authentication and especially authorization procedures are invoked on many operations. It may also be extended through policies, which are a set of rules defined by administrators to tailor the upper-level control on resources and testbeds usage.

3.3.7 Monitoring Functions

The monitoring functions are responsible for retrieving monitoring data for heterogeneous resources (e.g. compute, SDN, TN nodes) in the diverse testbeds of the federation, as well as aggregating and storing it. Monitoring data can be categorized in two types: *facility monitoring* and *infrastructure monitoring*.

The *facility monitoring* data encompasses the basic status information about the facility, such as the status of the servers' availability or network connectivity; and may also include the status of the functional components in the FELIX Control Framework, which is aggregated from RMs and ROs. This information can be offered through the user portal for both experimenters and administrators.

The *infrastructure monitoring* checks the status of the resources currently available or provisioned, as well as some data for the experiments. This includes availability of virtualization and networking hardware or other information such as uptime and resource usage statistics.

The slice monitoring developed in FELIX builds upon state of the art monitoring tools, as well as testbed specific (facility) monitoring tools. Among these tools we use SNMP and OpenFlow flow space monitoring statistics. Monitoring data, is provided not only graphically, but also via a Monitoring API so that the statistics can be directly used by control mechanisms to make routing decisions based on traffic measurements or link status. The use of monitoring data is key to implement reactive decisions in all use cases considered in the FELIX project, guaranteeing feedback to the Planning and Provisioning tools to ensure that the required slice resources are available and can be used for specific workloads.

3.3.8 User Access

The FELIX Graphical User Interface (GUI) offers intuitive access to the lifecycle management of an experiment for experimenters and general management operations for administrators. To do this, the User Portal communicates with the underlying modules of the FELIX architecture to use a subset of their functionalities to authenticate and authorize users, as well as track their actions and provide them with any requested resource management, operation or observation.

The experimenters are thus able to list the available resources, define a subset of resources and allocate or provision them, performs operational actions on the resources, retrieve a description of the resources made available to them as well as release the resources when no longer needed.

On the other hand, the administrators may configure and manage resources, define different types of policies, grant or revoke resource requests and monitor different sets of resources, among others.

4 The FELIX Use Cases

As previously mentioned, FELIX usage scenarios are clustered into two groups: Data Domain and Infrastructure Domain. Use cases in the Data Domain, include virtual infrastructure consisting of SDN islands interconnected with dynamic circuit-switched (inter-continental) networks. One important goal is to optimize the use of interconnectivity between testbeds to realize data migration. The Infrastructure Domain use cases describe user scenarios based upon federated resources placing emphasis on the optimized use of the infrastructure as a whole.

4.1 Data Domain Use Cases

Data Domain use cases are primarily oriented towards the efficient utilization of the physical network by taking advantage of SDN and NSI operations for the dynamic interconnection of testbeds dispersed across different continents. The focus here is on the coordination of caching, processing and network services rather than on the exact caching algorithms to be used, which are in the full scope of FELIX user/experimenter priorities and control.

The testbeds for the Data Domain use cases form a virtual infrastructure that consists of SDN islands (L2 domains) interconnected with dynamic circuit-switched networks (multi-domain transit networks). In this large-scale facility, data must be transferred from the origin to its destination end-point, typically in another SDN island. The following subsections summarize each use case and explain how the aforementioned flows of data traverse a real network.

4.1.1 Data-on-Demand: delivery of distributed data by setting data flows over the network

This use case investigates how to process large amounts of data stored in distributed sites. For instance, several applications, such as astronomical observations or collaborative investigations, generate huge data sets, typically stored in dedicated storage servers or devices in a nearby data centre. A user may want to run a post-processing algorithm on the data collected by different data providers. In this context, it is neither suitable nor efficient to move the whole data from the original sites to the final location but it could be convenient to perform sequential processing on partial data. Figure 4 depicts the main components of the scenario, the relevant actors and their potential interactions.

In this use case, an SDN-based (e.g. OpenFlow) controller maintains a global view of the whole network topology and could access monitoring information from the Data Centres (e.g. delay and jitter per link) then use this information at any given time to automatically select adequate paths based on any selected metric. Connectivity would be established by requesting the FELIX framework in each Data Centre thus effectively setting the SDN flowspace, inter-domain Transit Network connections, and stitching them together. By setting the most appropriate connection in each moment, an optimal use of the physical network resources could be achieved.

4.1.2 Data pre-processing for minimizing network latency effect for live data

This use case aims to provide near real-time data, e.g. satellite images, to users located in different and very distant places without incurring in the large Round Trip Delay (RTD) values typically found with transfers through the public Internet. Figure 5 presents an overview of this scenario. Common examples of such a condition are congested links during peak working hours of the day, which generate fast re-adaptations of the TCP transmission windows and, consequently, critical degradation of the throughput. This is particularly evident in inter-continental data transfers that typically have high values of RTD (>200ms) between the communicating servers due to long distance transmissions. In these situations, a dedicated platform would be placed near the receiver station and perform a suitable pre-processing of the data. This platform could be able to allocate computing, caching and networking resources at both source

and destination islands. It could also be able to implement on-demand and application-driven network services for the specific data transfers, which require well-defined network parameters. Consequently, this approach can significantly reduce the size of data to be delivered across the transit network and improve the overall system performance.

4.1.3 High-quality media transmission over long-distance networks

In the last few years we are experiencing a rapid evolution in media content delivery, especially in the context of the ultra-high definition of the video streaming, i.e. 4K and 8K resolution. This evolution directly relates to a higher quality of media playback, but also imposes higher bandwidth and lower delay constraints on the network. In this scenario, illustrated in Figure 6, hardware optimization is required for the transmission and reception of the data content, especially in a very long-distance environment.

At the same time, network streamlining is needed both in the transport segments and in the inter-data centre networks (NSI- and SDN-enabled). In this use case, all the defects of poor management and control of the network will manifest themselves in visible playback artefacts: jitter, incorrect frame sequencing, transmission disruption, etc. Moreover, strict requirements are imposed in order to serve 3D video to the user, as two flows have to be delivered separately for the left and right eye. In this scenario, proper synchronization is extremely important to achieve a satisfactory quality of service. This is measured through Quality of Service (QoS) and Quality of Experience (QoE) metrics.

4.2 Infrastructure Domain Use Cases

The Infrastructure Domain use cases are mainly concerned with the services and workloads that can be facilitated by a software platform built on top of the federated resources. It is important to note that both the Infrastructure and Data Domain use cases share common architectural, trust and security assumptions. In the Infrastructure Domain use cases, we consider network, computing and storage resources that can dynamically migrate over the allocated physical environment. This work is in line with recent developments in leading standardization fora, such as IETF and ETSI, where significant attention has been drawn from both industry and academia towards network service chaining and the ability to relocate network functions, infrastructure scale-out and scale-up, as well as continuous service delivery [19]. The following subsections introduce the Infrastructure Domain use cases and explain how the services can be deployed in a large-scale facility, such as FELIX.

4.2.1 Inter-cloud use case: data mobility service by SDN technologies

This use case focuses on cloud systems and the services provided by them in carrier-grade, mission-critical areas. This includes electronic administration, medical care and finance. To satisfy the requirements, these complex cloud systems should meet demands of an end-to-end guaranteed service quality, reliability of compliance and energy efficiency. In this context, every single-cloud system is limited by its available resources. This limit can be easily exceeded with a flexible reassignment of resources belonging to different cloud systems. Therefore, it is important to establish cooperation between data centres, at least on a temporary basis.

For example, consider a user who moves to a remote location due to a business trip. The user wants to use a number of cloud-based services with the same level of quality of experience as when using local resources and on par with the quality experienced in their home network. Note that in this case, traditional mobility management solutions [20] would not be able to mitigate the expected large propagation delays between the present user location and the data centre processing the user's workload. Instead, the scenario illustrated in Figure 7 shows that it would be preferable to transfer user data (such as credentials, applications and services) to a cloud system closer to the visiting place (e.g. the cloud with minimum delay relative to the user's visiting place); and therefore reconstructing the user's work place in a remote location.

4.2.2 Follow the sun (or moon) principles

As detailed in [21], Internet usage curves follow a similar daily pattern everywhere in the world, and there is a natural shift in the load of data centres to places in the world where it is currently daytime. The opposite is true during the night, when data centres are under a different amount of load. This is often referred to as the 'follow the sun/moon' principle. Moreover, the prices of renewable energy strongly depend on the availability of wind and solar energy (green energy). As a result, several data centres are moved to locations such as Iceland and Finland and perhaps in the future to desert areas.

In this case, one could shift the load of one data centre to another one following two different approaches (Figure 8): a) move the entire workload to a more efficient data centre basically with a re-routing of the user's traffic, or b) handle the user's requests at the less efficient data centres by delegating the work-flow to more efficient data centres. It is important to note that both scenarios require dynamic and on-demand end-to-end connections between the federated data centres. Moreover, when the workload is moved from one data centre to another, a number of different resources (network, compute and storage) need to be configured accordingly.

4.2.3 Disaster recovery by migrating IaaS to a remote data centre

This use case is inspired by the Business Continuity Planning (BCP) of key services to cloud providers. This is particularly pertinent after the experience of the Great East Japan earthquake in 2011. Typically, the cloud systems are managed by Infrastructure as a Service (IaaS) software, such as OpenStack or CloudStack, and provide isolated tenants on physical resources (computers, storage and network) in a data centre with multiple IaaS users. These users expect a stable and fault-free environment, but under particular conditions, in a serious disaster, it can be difficult to continue providing the desired services. In such a case, middleware can assist in enabling the migration of the cluster of servers and virtual machines to some remote data centre and guarantee business continuity. Another factor used in creating this use case is the Hardware as a Service (HaaS) paradigm [22], which can dynamically configure virtual IaaS-enabled resources using nested virtualization technologies (e.g. KVM and FlowVisor). These resources can be migrated on the HaaS layer of another data centre, as depicted in Figure 9, coordinating the configuration of the hypervisor resources with the network bandwidth constraints to allow a fast and efficient migration of the IaaS instance from one site to another.

5 Conclusions and Future Work

We presented the FELIX Control Framework architecture and six use cases for large-scale SDN experiments over cross-continental federated environments. We grouped the set of use cases into two major categories, namely the Data Domain and Infrastructure Domain, in order to better reflect their primary applicability area and stakeholders. These scenarios highlight the necessity to have a single management and control of the intra- and inter- connectivity for the data centres. These scenarios serve as the foundation for the development of complex architectural models and software platforms able to manage resources in more efficient ways.

From the users' perspective, all presented use cases apply to the same and unique FELIX framework architecture that includes common system functionalities derived from the specific use cases and the users' goals, in the form of requirements. The current list of use cases is not meant to be exhaustive.

The resulting architecture allows experimenters of the testbed to request, manage, and monitor a slice in a heterogeneous, distributed, and multi-domain environment. This high-level specification is generic enough to allow flexible and scalable deployments in the different testbeds that are part of the federated environment.

The work in the FELIX project is now proceeding towards the implementation of the FELIX system components. As part of our future work, we aim to complete the development of the feature set defined for each software component and – in parallel – test, validate and refine the implemented functionalities through the use cases presented in this paper.

6 Acknowledgements

This work was conducted within the framework of the EU-Japan FELIX project, which is partially funded by the Commission of the European Union and the National Institute of Information and Communications Technology (NICT), Japan. Study sponsors had no role in writing this report. The views expressed do not necessarily represent the views of the authors' employers, the FELIX project, the Commission of the European Union, or the National Institute of Information and Communications Technology (NICT), Japan.

7 References

[1] FELIX Project website: http://www.ict-felix.eu

[2] L. Peterson, R. Ricci, A. Falk and J. Chase, Slice-based Federation Architecture (SFA) v2.0, July 2010.

[3] GENI Aggregate Manager API. <http://groups.geni.net/geni/wiki/GeniApi>.

[4] R. Krzywania, W. Bogacki, B. Belter, K. Pentikousis, T. Rothe, G. Carrozzo, N. Ciulli, C. Bermudo, T. Kudoh, A. Takefusa, J. Tanaka and B. Puype, *Experiment Use Cases and Requirements*, FELIX Deliverable D2.1, September 2013. Available at http://www.ict-felix.eu.

[5] R. Krzywania, W. Bogacki, B. Belter, K. Pentikousis, T. Rothe, M. Broadbent, G. Carrozzo, N. Ciulli, R. Monno, C. Bermudo, A. Vico, C. Fernandez, T. Kudoh, A. Takefusa, J. Tanaka and B. Puype, *General Architecture and Functional Blocks*, FELIX Deliverable D2.2, February 2014. Available at http://www.ict-felix.eu.

[6] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, J. MacAuley and C. Guok, *NSI Connection Service v2.0*. May 2014. Available at http://www.ogf.org/documents/GFD.212.pdf.

[7] I. Monga, E. Pouyoul and B. Tierney, *Dynamic creation of end-to-end virtual networks for science and cloud computing leveraging OpenFlow/Software Defined Networking* in *Proceedings of the 2012 TERENA Networking Conference (TNC'12)*, May 2012.

[8] I. Monga, E. Pouyoul and C. Guok, *Software Defined Networking for big-data science*, in *Proceedings of the 2012 High Performance Computing, Networking, Storage and Analysis Conference (SCC'12),* November 2012.

[9] A. Sadasivarao, S. Syed, P. Pan, C. Liou, A. Lake, C. Guok and I. Monga, *Open Transport Switch: A Software Defined Networking Architecture for Transport Networks* in *HotSDN workshop in ACM Special Interest Group on Data Communication (SIGCOMM'13)*, August 2013.

[10] Apache CloudStack documentation. *Cloud Infrastructure Concepts*. <http://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.1.0/html/Admin_Guide/cloud-infrastructure-concepts.html>.

[11] Scaling Openstack. <http://docs.openstack.org/openstack-ops/content/scaling.html>.

[12] M. Suñé, L. Bergesio, H. Woesner, T. Rothe, A. Köpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda, M. Kinde, T. Dietzf, A. Autenriethg, V. Kotronis, E. Salvadori, S. Salsano, M. Körner, S. Sharma, *Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed*, *The International Journal of Computer and Telecommunications Networking*, December 2013.

[13] T. Salmito, L. Ciuffo, I. Machado, M. Salvador, M. Stanton, N. Rodriguez, A. Abelem, L. Bergesio, S. Sallent, L. Baron, *FIBRE - An International Testbed for Future Internet Experimentation* in 32º *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'14)*, 2014.

[14] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, M. Sioutiss, A. Quereilhac, Y. Al-Hazmi, F. Lobillo, F. Schreiner, C. Velayos, A. Vico-Oton, G. Androulidakis, C. Papagianni, O. Ntofon, M. Boniface, *Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities* in *Proceedings of Future Network & Mobile Summit Conference*, 2013.

[15] A. Takefusa, H. Nakada, T. Kudoh, Y. Tanaka and S. Sekiguchi, *GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources*, *Lecture Notes, Computer Science of the2008 Job Scheduling Strategies for Parallel Processing (JSSPP'08),* April 2008, vol.4942, pp. 152-168.

[16] *Y. Kanaumi, S. Saito, E. Kawai, S. Ishii, K. Kobayashi and S. Shimojo, RISE: A Wide-Area Hybrid OpenFlow Network Testbed,* IEICE Transactions on Communications, January 2013, vol. E96-B, no. 1, pp. 108-118

[17] U. Toseef, A. Zaalouk, T. Rothe, M. Broadbent and K. Pentikousis, *C-BAS: Certificate-based AAA for SDN Experimental Facilities* in *Proceedings of the 2014 European Workshop on Software Defined Networking (EWSDN'14),* September 2014.

[18] *Common Federation API for any GENI-compatible federation.* November 2013. <http://groups.geni.net/geni/wiki/CommonFederationAPIv2>.

[19] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert and C. Meirosu, *Research Directions in Network Service Chaining* in *Proceedings of 2013 Future Networks and Services (IEEE SDN4FNS'13)*, November 2013.

[20] K. Pentikousis and P. Bertin, *Mobility Management in Infrastructure Networks*, *IEEE Internet Computing*, September 2013, vol.17, iss. 5, pp. 74-79.

[21] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag and B. Maggs, *Cutting the Electric Bill for Internet-scale Systems* in *Proceedings of the 2009 ACM Special Interest Group on Data Communication (SIGCOMM'09)*, August 2009.

[22] R. Takano, A. Takefusa, H. Nakata, S. Yanagita and T. Kudoh, *Iris: Inter-cloud Resource Integration System for Elastic Cloud Data Center* in *Proceedings of the 2014 International Conference on Cloud Computing and Services Science (CLOSER'14)*, April 2014.

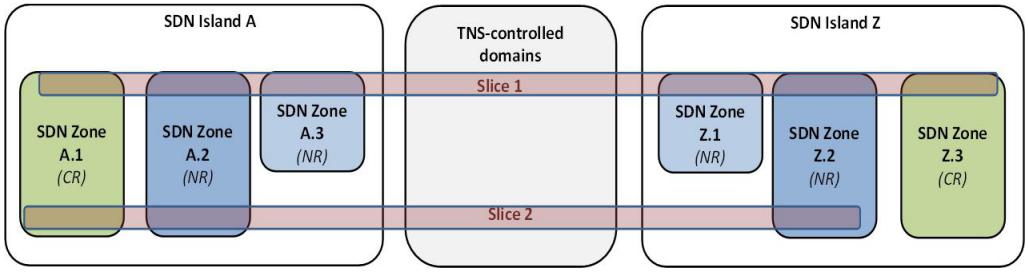


Figure 1. FELIX key concepts: transit domains, islands, zones and slices.

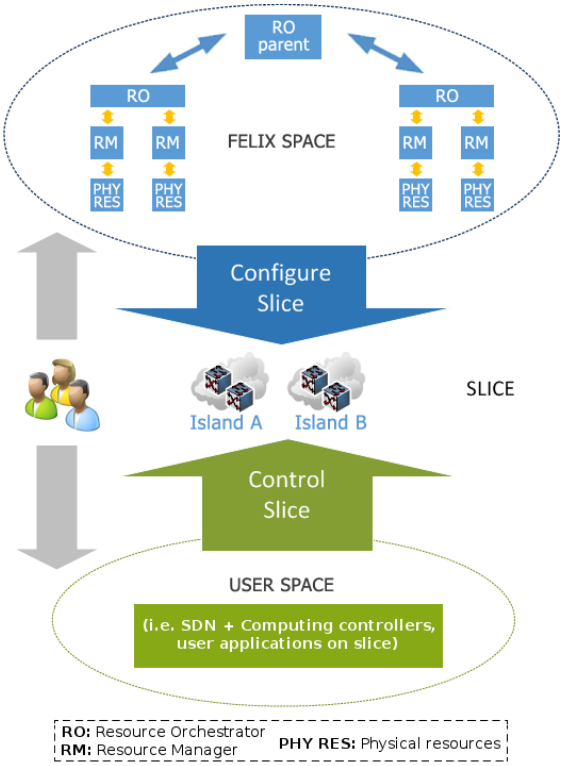
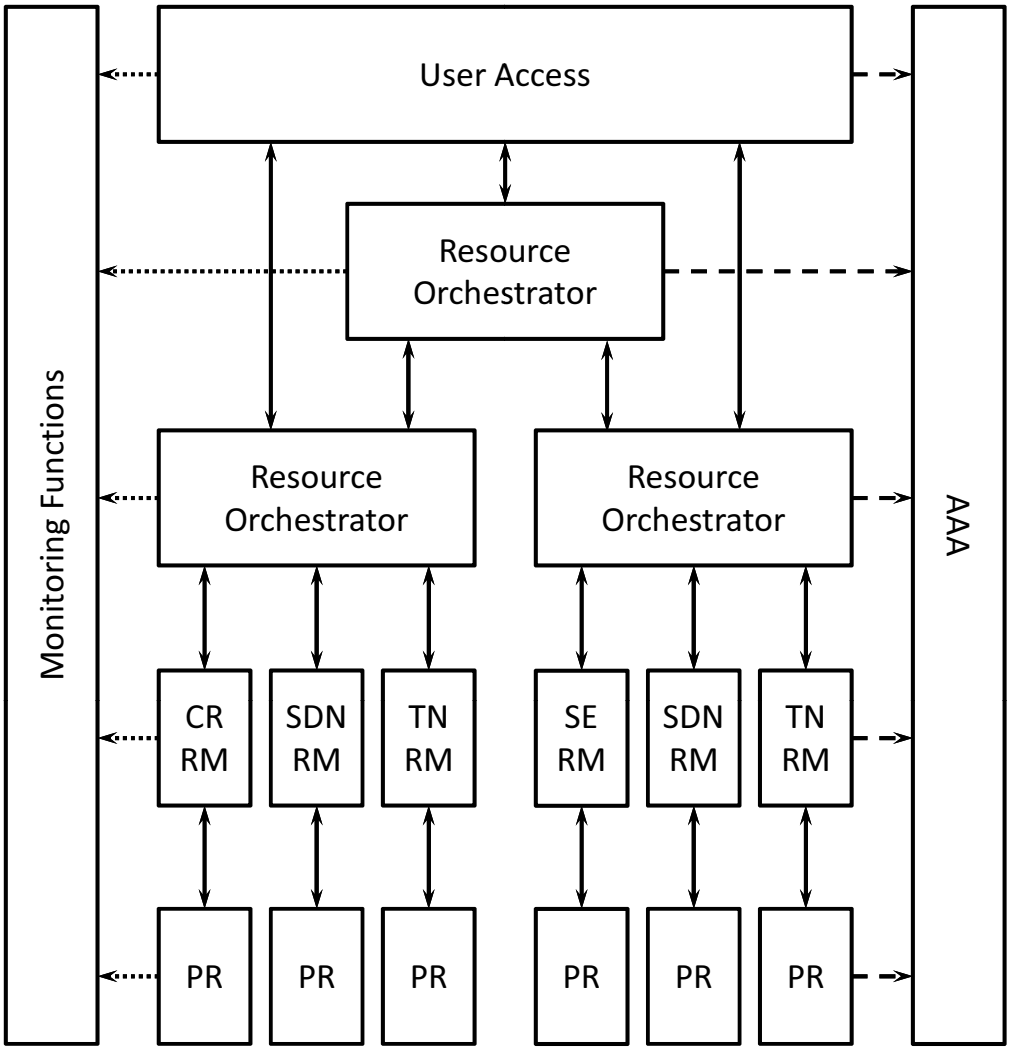


Figure 2. FELIX Architecture and Spaces.



AAA: Authentication, Authorization & Accounting
CR: Computing Resource
PR: Physical Resource
RM: Resource Manager
SDN: Software Defined Network
TN: Transit Network
SE: Stitching Entity

Figure 3. Building blocks of the FELIX Architecture.

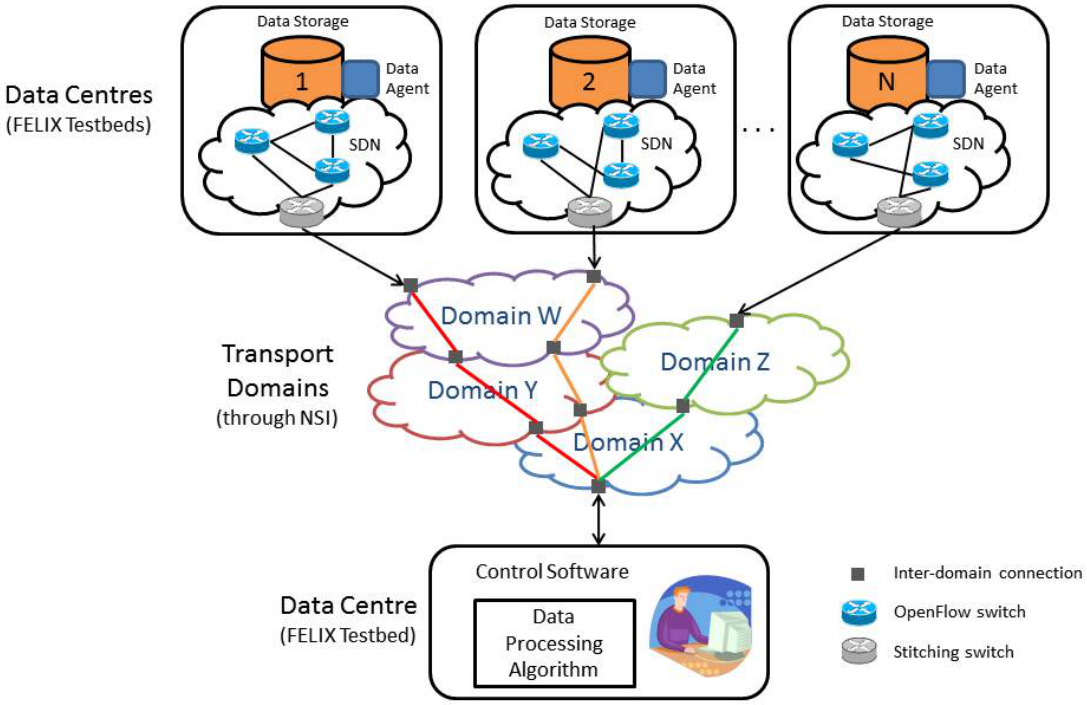


Figure 4. Data-on-demand: distributing data via programmable data flows.

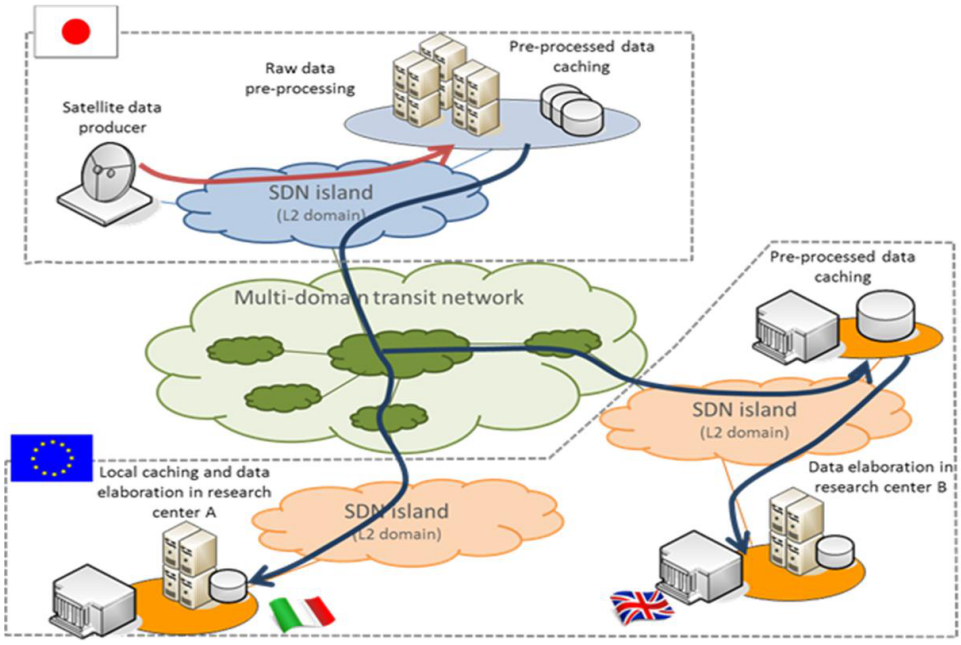


Figure 5. Data pre-processing: minimizing network latency effect for live data.

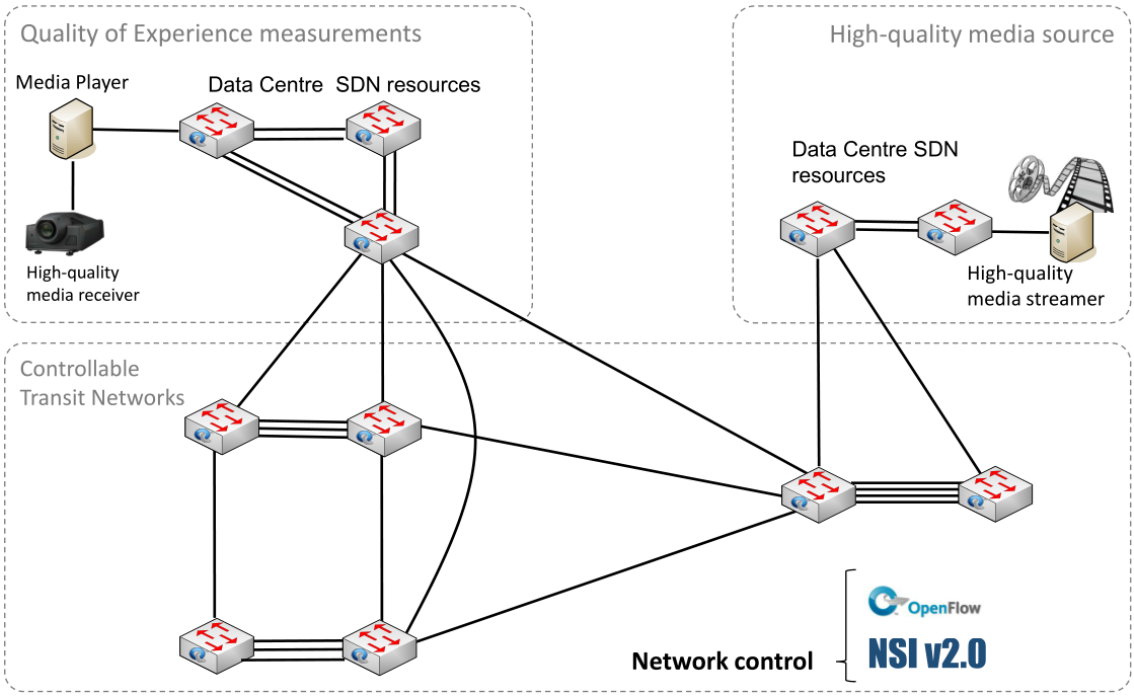


Figure 6. High-quality media transmission.

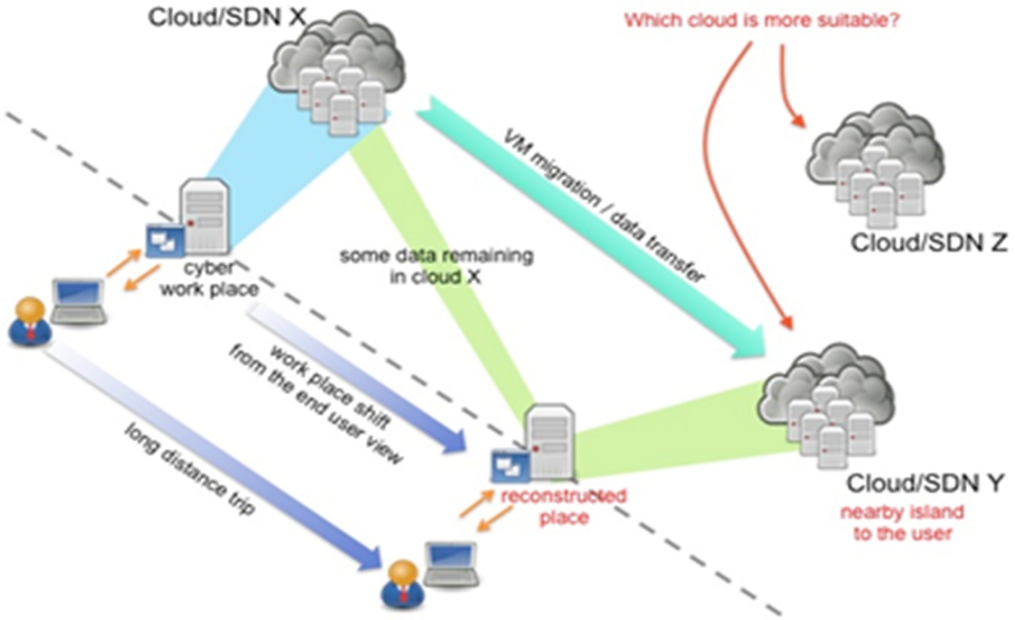


Figure 7. The inter-cloud data mobility scenario.

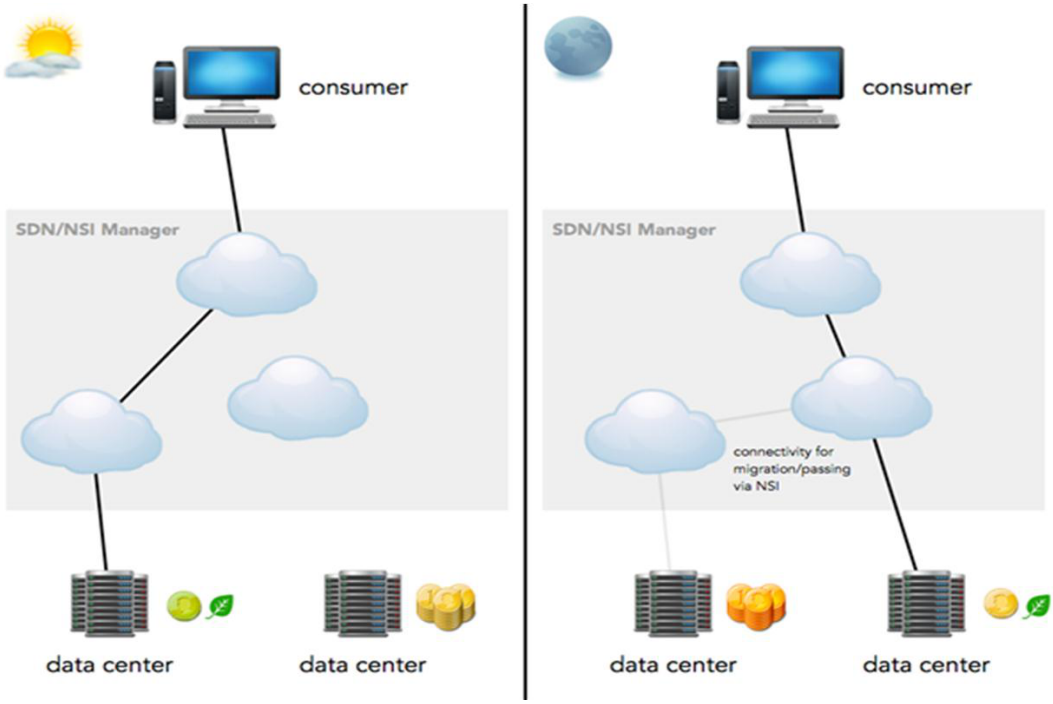


Figure 8. The 'follow the sun-moon' use case.

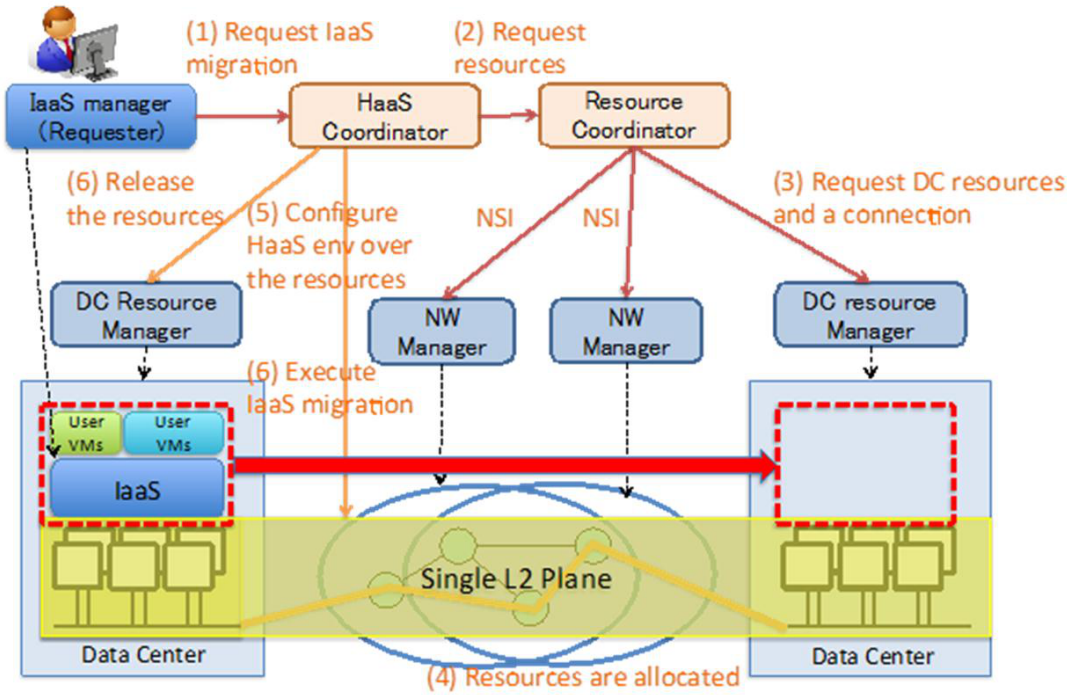


Figure 9. Disaster recovery through IaaS migration in remote data centre**.**