FEDERATED TEST-BEDS FOR LARGE-SCALE INFRASTRUCTURE EXPERIMENTS
FELIX EU-JP

Collaborative joint research project co-funded by the European Commission (EU) and National Institute of Information and Communications Technology (NICT) (Japan)

Grant agreement no:	608638
Project acronym:	FELIX
Project full title:	"Federated Test-beds for Large-scale Infrastructure eXperiments"
Project start date:	01/04/13
Project duration:	36 months

Deliverable D2.1
Experiment Use Cases and Requirements

Version 1.0

Due date:	30/09/13
Submission date:	30/09/13
Deliverable leader:	PSNC
Author list:	Radek Krzywania (PSNC), Wojbor Bogacki (PSNC), Bartosz Belter (PSNC), Kostas Pentikousis (EICT), Tom Rothe (EICT), Gino Carrozzo (NXW), Nicola Ciulli (NXW), Carlos Bermudo (i2CAT) , Tomohiro Kudoh (AIST), Atsuko Takefusa (AIST), Jin Tanaka (KDDI), Bart Puype (iMinds)

Dissemination Level

☒	**PU:**	Public
☐	**PP:**	Restricted to other programme participants (including the Commission Services)
☐	**RE:**	Restricted to a group specified by the consortium (including the Commission Services)
☐	**CO:**	Confidential, only for members of the consortium (including the Commission Services)

Abstract

This document presents initial set of FELIX user scenarios alongside the corresponding user requirements for the FELIX virtual infrastructure and overall system. Six use cases have been detailed, further grouped into two major groups (Data Domain and Infrastructure Domain) to better reflect their primary applicability area and stakeholders. The user scenarios identified in the document will serve as a foundation for defining the FELIX architecture.

Project:	FELIX (Grant Agr. No. 608638)
Deliverable Number:	D2.1
Date of Issue:	30/09/13

Table of Contents

Figure Summary

Table Summary

Executive Summary

The FELIX Work Package 2 (WP2) main objective is the development of an architecture framework for the federation of Future Internet testbeds. The project work plan assumes a two-stage development process for the final FELIX architecture. In the first stage, requirements for the architecture are collected and analysed. The requirements identified within the project are represented in a form of FELIX user scenarios. In the second stage the collected requirements will be translated into a detailed specification of functional components for the project architecture alongside the corresponding interfaces.

This deliverable reports on FELIX usage scenarios, clustered around two groups: **Data Domain use cases** and **Infrastructure Domain use cases**. The Data Domain use cases in FELIX consider a virtual infrastructure that focuses on the efficient use of software-defined networking for dynamically and efficiently interconnecting geographically dispersed testbeds across two continents. In scope of the Data Domain use cases the virtual infrastructure will consist of SDN islands interconnected with dynamic circuit-switched (inter-continental) networks. An emphasis and goal is to optimize the use of interconnectivity between testbeds to realize data migration. On the other hand, the Infrastructure Domain use cases describe user scenarios of the virtual infrastructure based upon federated resources with an emphasis put on the optimized use of the infrastructure as a whole, including the migration of entire workloads of data processing.

1 Introduction

The document introduces the FELIX Use Cases and User Scenario Requirements, which will serve as a foundation for defining the FELIX architecture. The document does not define the architecture itself, yet it specifies some functionalities that are required. This deliverable follows the terminology of RFC2119 [1] with respect to the use of the capitalized words MUST, SHOULD, and MAY in phrases which indicate the corresponding user scenario requirements which will define particular features of the FELIX framework. Note that the list of user scenario requirements in this document will serve as the foundation for the architectural requirements which will follow in the upcoming deliverable D2.2. This document presents the results of our early-stage effort to structure what FELIX products will do exactly at the end of the project, based on realistic user scenarios as defined by the consortium partners as a whole.

Chapter 2 of this deliverable presents the user scenarios which can be used for FELIX framework validation and demonstrations. The chapter also explains the use case description methodology adopted in FELIX, and details six user scenarios. The Chapter 3 of this deliverable summarizes in a concise manner the first draft list of the FELIX framework requirements, as derived from the user scenarios. Each of the requirements contains a short description, explaining its purpose and the associated features. Finally, Chapter 4 contains summary conclusions and indicates further directions of efforts undertaken within the FELIX project.

2 FELIX User Scenarios

At the core of the FELIX project objectives lies the definition of a common framework for federated Future Internet (FI) testbeds, which are dispersed across two continents. This framework will enable its user to a) request and obtain resources across different testbed infrastructures dynamically; b) manage and control the network paths connecting the federated SDN testbed infrastructures; and c) use distributed applications executed on the federated infrastructures.

In this section we introduce the rationale behind the use cases we consider in the project as well as the requirements that stem from them and can thus guide the FELIX framework development. We start this exposition in the following subsection by providing a definition of the resources available in a FELIX testbed federation. We then describe the FELIX vision on the two top-level domains we consider in our work (Section 2.2), followed by the key system concepts that will be in the scope of design and deployment in our target architecture (Section 2.3). In Section 2.4 we describe the methodology employed to define our use cases, while from Section 2.6.1 onwards we define in detail the FELIX user scenarios which we will use in our future work in the project.

2.1 FELIX Federated Resources

The term “resources” in the context of FELIX includes networking, computing, and storage capacities available at geographically dispersed facilities under the administrative control of different but cooperating stakeholders. Taken as a whole, the FELIX federated resources create a virtual infrastructure which 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. a cloud operators’ data centers; and b) loosely coupled, interconnected islands which allow for remote access to certain resources. Today, one can consider scenarios where, for example, scientific data is obtained and stored in one continent and made available to other researchers around the world. In this case, “resources” would include the original location of data storage, local resources for processing and storing the incoming data at the researcher’s institution, and the end-to-end network connectivity. Scenarios like this are today implemented using remote access and file transfers over TCP/IP via the global Internet infrastructure. However, this is not necessarily the most effective use of all resources available.

In FELIX we consider the development of a framework that can outperform current best practices and use the federated resources better. In order to design such a framework we need to identify the requirements stemming from i) the end user, ii) facility capabilities, and iii) network operator perspective.

First, the end user perspective includes features such as ease-of-use, extending the researchers' capacity to do research using federated resources and thus, for instance, expanding the reach of their experimental analysis, and enhancing scientific collaboration across the federated institutions. We are basically interested in network enablers that increase researchers' productivity and make better use of the federated infrastructure resources. Note that since FELIX is fundamentally based on European and Japanese research testbeds and networks, we consider researchers as a first type of user for the FELIX framework. However, the design which will commence once the requirements are clarified will not aim to create a framework suitable solely for a particular type of users. On the contrary, the mechanisms developed will be generic enough to be applicable to a wide range of user types and interests.

Second, facilities capabilities include what we could refer to as the "cloud view", i.e. making the most of local and remote resources, through the establishment of a common and shared (according to policies) virtual infrastructure overlaid on multiple network domains. In this case, one would be interested in tools and mechanisms that allow researchers, for example, on the one hand to discover, request and obtain resources, and on the other hand manage and monitor their provisioned resources dynamically. For example, statistical analysis of the processed data may point to an unforeseen direction for further analysis. Researchers may be able to dynamically change the provisioned resources (e.g. request more computing resources elsewhere in the virtual infrastructure, and either transfer data dynamically or entire workloads) to follow, so-to-speak, new leads in a much shorter time spans than what is possible today with the prevailing batch modus operandi.

Third, we need to consider how to optimize the network resources, as FELIX is primarily a network-oriented testbed project. In this case, the network operator could refer to the federation of research networks and research testbeds, as well as to a cross-continental network operator with trusted relationships with other operators. We do not differentiate in FELIX on whether the operator is commercially oriented or not. Instead we focus on the network enablers that we perceive as necessary, and how they can be enhanced further in order to serve the goals of FELIX. For example, the current "one-size fits all" approach using always-on, end-to-end TCP/IP connectivity limits our capacity to make the most of novel access technologies. TCP/IP is excellent for remote access to resources, but it cannot make the most of access underlays based on the latest technologies. TCP/IP-based remote access also does not provide resource isolation, the possibility to dynamically take in a large amount of dedicated network capacity, let alone multi-point communication. In FELIX we aim to address these problems through the dynamic establishment and tear-down of network flows (based on L2 switching and L3+ routing/forwarding) which enable finer control granularity of the network.

2.2 Towards a Virtual Infrastructure through Federation

As discussed in ECSEL [2], connecting facilities at continental, let alone inter-continental scale as envisioned in FELIX is, to say the least, challenging. Monga et al. motivate the need for the ability to connect facilities similar to the FELIX testbeds at the lower layers (e.g. L2) and avoid the system overheads that doing so at the TCP/IP layers would introduce. Unfortunately, the proposal in [2] reports early stage work and does not follow a de-facto standard for R&E networks such as NSI. Moreover, their treatment of what is needed inside each island from a network control point of view is less than complete. Similarly, the authors do not discuss policies and trust in their line of work. We believe that all these aspects will play a crucial role in the adoption of framework suitable for federated resources and we find the current literature lacking a solution for the purposes of the targeted cross-continental federation in FELIX.

That said, the present analysis of the latest research literature on this topic does indicate that we will need to introduce new APIs and logic for globally distributed heterogeneous facilities (e.g. OFELIA islands and JGN-X RISE testbeds) which will capitalize on SDN and NSI mechanisms and protocols and allow for dynamic, on-demand establishment of end-to-end cross-continental virtual network infrastructures. In short, we consider that it is feasible to use SDN mechanisms, such as OpenFlow control, in the local infrastructure coupled with WAN-based mechanisms such as NSI, possibly extended in FELIX, to create a virtual networked infrastructure.

With the virtual network infrastructure in place, we can then interchange resource information, share overall resource pools, and provide dynamic network interconnectivity between and within islands. The user is expected to see his experiment slices as an integrated whole, despite of using resources from different management frameworks and from distant geographic locations.

In order to proceed with the architectural definition, we need to take into consideration the end-user experience in real-world use cases, which will form the base out of which we can obtain detailed answers on requirements for the control framework for federated SDN infrastructures. As we get more technical, these user scenarios can show us the way on how to integrate virtual slices globally, which architecture interfaces could be needed, what components need to be defined or extended, and finally define the workflows of infrastructure management over the general architecture. Finally, we also point out that the overall user scenarios will serve as a storyline upon which we can validate the project results.

While SDN testbed infrastructures are constructed from the view point of network research and development, it goes without saying that computing and storage resources are also important components of each testbed, and some FI usage will mainly use network resources to move data, and some will use infrastructure including computing and storage resources to provide ICT services. Therefore, as detailed later in this document, we consider two major classes of use case scenarios for the virtual infrastructure based on federated testbed resources. Namely, we will refer to the first category of use cases as the “data domain use cases” since the primary focus is the use of data. We will refer to the second category as the “infrastructure domain use cases”, which includes all three factors in the testbeds: networking, storage and computing resources. The data domain use cases focus more on network usage, and as examples, we chose data delivery, data workflow management, and establishing a dynamic network path for streaming as the use cases. The infrastructure domain use cases focus on management of virtual infrastructures and consider all three resource types.

2.3 Key FELIX System Concepts and Definitions

Before digging into the specific aspects of the two major categories of user scenarios introduced above, it is important to discuss some key system concepts and definitions that are common across the board in the FELIX user scenarios. These concepts are derived from the discussion and agreements reached between the project partners on the major roles and functions required in (and expected from) the FELIX architecture. Currently these concepts have the objective to act as a “straw man” of the high level framework for the FELIX system, mostly for the purposes of a common background for the execution of the user scenarios. Our subsequent work on consolidating the FELIX user scenarios and requirements will allow us to further refine and detail these concepts, thus turning them into the appropriate FELIX architecture elements.

FI experimental facilities (or SDN-controlled network domains). The facilities are the experimental facilities integrated in the FELIX infrastructure. All FI facilities are controlled by dedicated software, which exposes interfaces which can be used by a federation framework to orchestrate resources in a multi-domain environment. The SDN controlled network domains are illustrated is Figure 1.

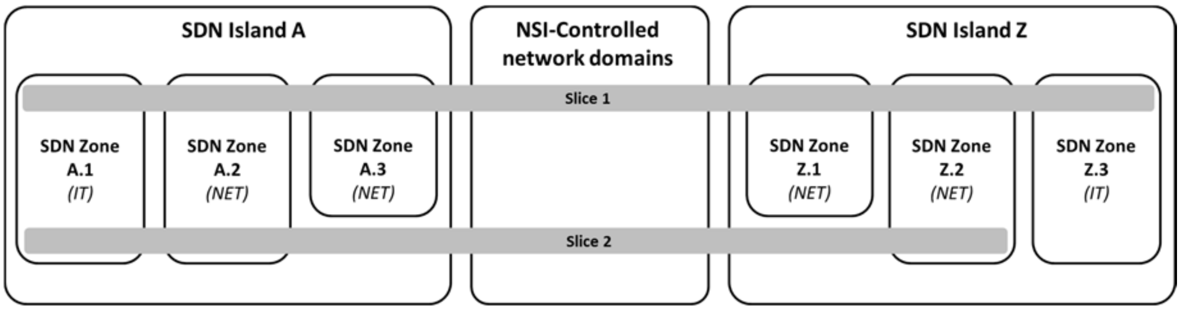


Figure 1. FELIX infrastructure key concepts

SDN Island. A set of virtualized network and IT resources under the same administrative ownership/control. An SDN island could consist of multiple SDN zones, each characterized by a specific set of control tools & interfaces.

SDN Zone. A set of resources grouped for homogeneity of technologies and/or control tools and/or interfaces (e.g. L2 switching zone, optical switching zone, OF protocol controlled zone, other transit domain zone with a control interface). The major goal of defining SDN zones is to implement appropriate policies for the increasing availability, scalability and control of the different resources of the island. Examples of zone definitions are available in popular Cloud Management Systems (CMS) like Cloudstack (e.g. refer to the Cloudstack Infrastructure partitioning into regions, zones, pods, etc., [3]) and OpenStack (e.g. refer to the infrastructure partitioning in availability zones and host aggregates [4]).

NSI-controlled network domains. The network domains use the Network Service Interface to expose automatically and on-demand control of the connectivity services and, optionally, inter-domain topology exchange. In order to federate resources belonging to distant experimental facilities it must be ensured that interconnectivity is provided on-demand and with a specific granularity. 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.

Slice. A user-defined subset of virtual networking and IT resources, created from the physical resources available in federated SDN Zones and SDN Islands. An SDN Slice has the basic property of being isolated from other slices defined over the same physical resources, and being dynamically extensible across multiple SDN Islands. Each SDN Slice instantiates the specific set of control tools of the specific zones it traverses.

SFA-based federation. The minimal set of interfaces and data types that permit a federation of slice-based network substrates to interoperate (Slice-Based Federation Architecture). SFA is the “de-facto” standard for the testbed federation. In FELIX, it is assumed the use of SFA to provide a federation framework between the existing testbed management platforms deployed at partners’ premises.

FELIX SDN Controller. The functional element that can control the virtual network resources of the FELIX federated infrastructure. The FELIX SDN Manager will play a role of intermediate component between internal architecture components and external network management platforms. The SDN Manager will control the use of resources in the access domains (e.g. OpenFlow-based L2 resources) and the NSI in transit domains in order to provide a multi-domain network provisioning framework. Therefore, the SDN Controller manages resource provisioning, resource maintenance/monitoring (south bound) and inter-operation with peering controllers (east-west bound).

FELIX Resource Orchestrator. The functional element that can orchestrate the reservations of network (and possibly IT) services in the FELIX federated infrastructure, assuring that all required resources are provided at specified moment in time (e.g. SDN, network, computing, and storage resources) at particular locations. In order to manage resources of different experimental facilities, a common description language must be introduced. In FELIX, federated testbed resources are described through the use of RSpecs (the Resource description language of SFA). Existing RSpecs for equivalent resources (e.g. developed in the projects OFELIA, FIBRE, FED4FIRE) are used as baseline in FELIX, and they will be adapted and extended when necessary. The "Resource discovery" functionality is key input to the resource orchestration, as it makes available new testbed resources to the federation. Similarly, "Resource monitoring" is vital to provide information about the actual status of resources available in the federation.

FELIX Experiment Control and Management. The functional entity that provides mechanisms for the creation of experimental slices across Europe and Japan for end users. These mechanisms involve interaction with the resource orchestration, in order to create the allocation of available resources into the user slice.

FELIX Monitoring Framework. The framework for managing the monitoring functions on slices and users' experiments. FELIX will design a monitoring framework into SFA to retrieve, aggregate and store the monitoring data of the slices containing resources of multiple testbeds. Key functions of this monitoring framework include:

- monitoring of resources available at each FELIX testbed island;
- monitoring of dynamic connectivity provisioned by the NSI Connection Service (NSI-CS).

The first aspect, i.e. monitoring of resources in islands, covers an aggregation of monitoring data from different monitoring instances at each of the islands. The second aspect, i.e. transit network connectivity service monitoring, deals with integrating the existing connectivity status enquire capabilities exposed by the NSI with the overall resource monitoring of slices and experiments. The Slice and Experiment Monitoring component will expose relevant interfaces towards the FELIX resource orchestration to allow a provisioning of resources actually available for end users.

2.4 Use Cases Description Methodology

The FELIX user scenarios described in this chapter have been identified with the primary objective of motivating concepts and innovations expected through FELIX, in order to address existing issues or barriers related to the internetworking and the allocation of resources among very distant locations (e.g. Europe, Japan). Also, these use cases have been selected to better highlight the functional and non-functional requirements of the FELIX architecture and, consequently, the system/platform to be designed and implemented.

There is no standard template for composing use cases and, to our knowledge, no way of defining a single template that is effective for all application domains. Nevertheless, the FELIX partners have decided to adopt and slightly enhance the use case template used in the FP7-ICT GEYSERS project [5]. The FELIX consortium is convinced that this methodology allows us to better describe the user scenarios, shedding more light into the common aspects as well as the differences between the scenarios defined, while satisfying the objectives above. The approach also covers some general guidelines available in the literature [6].

The FELIX user scenarios template consists of three parts:

- The first part describes the elements of the use case required to understand its key aspects. This includes the storyline, goals, actors, components, preconditions, triggers and post-conditions.
- The second part of the template provides further details about the use case: these details describe the expected system behaviour to implement the use case. It is suggested that these are presented in the form of activity and state diagrams.
- The third section of the template provides further information that motivates the use case and gives hints towards the FELIX requirements.

Use case	
Number	<a sequential reference number>
Scenario / use case name	<short name of the scenario / use case>
Storyline	<describe the storyline in a few sentences – *up to 2000 characters* > short description of the use case essentials just to give an overview of the use case purpose and problem statement. This should not be very technical, able to read by people out of the project. The character limits is not obligatory, but is a suggestion to keep this chapter short, thus anyone can get the idea out of it in relatively short time.
Goal (s)	< the functionality or behaviour that is expected to be provided by the system once the use case is executed >
Figure to visualize the use case/scenario	<a picture to capture the main workflow concepts>
Actors	< the individuals and their means that trigger system reactions *,* e.g. island owner/infrastructure provider in JP and EU, experimenter, etc.>
System components	<the macro FELIX system entities expected to be involved in the execution of the use case and which role/function may be needed in > **FI experimental facilities** **NSI-controlled network domains** **SFA-based federation** **SDN Manager** **Resource Management** **Experiment Control and Management**

	Monitoring Framework
Preconditions	< those actions that must occur before the execution of the use case in order to obtain the described behaviour >
Trigger	< the main action that starts the use case execution >
Post conditions	< those conditions that could occur once the use case has been executed and the system continues its operations > It should explain when the execution of a use case scenario can be considered as a success or failure. The section should be short and not extremely technical. **Success conditions:** < which conditions demonstrate the correct and successful completion of all the steps in use case > **Failed End protection:** <which conditions/actions should occur to keep everything consistent, e.g. do not have orphan connections installed in NSI domain, etc. >
Detailed Steps	
State Chart or Activity Diagram Explaining the steps in the scenario	<a picture to capture the main states and activities related to the execution of the use-case>
Explanation of "states" and "transitions" in state chart or "activities" and "flows" in activity diagram	<phases are identified for the use-case>

Table 1: FELIX Use case template

2.5 Data Domain Use Cases

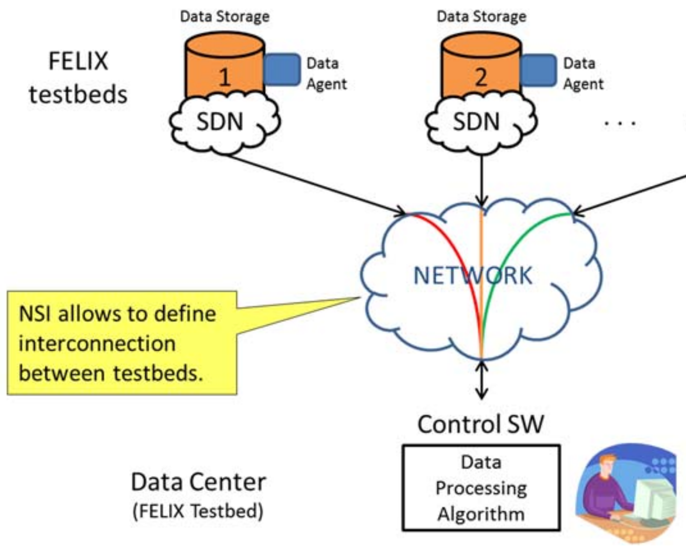
The Data Domain use cases in FELIX consider a virtual infrastructure that focuses on the efficient use of software-defined networking for dynamically and efficiently interconnecting geographically dispersed testbeds across two continents. The virtual infrastructure will consist in these use cases of SDN islands interconnected with dynamic circuit-switched networks. Although we do not claim that the uses cases presented in this document are exhaustive, we do believe that they are very much representative of the real-world use cases and include all the salient features that can drive the architectural discussion which will follow the use case specification.

As we document below, we consider three particular data domain use cases which focus on data delivery, data workflow management and data streaming. In all cases data are transferred from their source of origin to data destinations in other SDN island(s). Note that the use cases do not assume two end points in the communication. On the contrary the use cases consider the possibility for having many-to-many connections, either because the end user wants to put together data from a variety of sources, or because the management infrastructure understands that better resource utilization is possible through multipoint communication. In these use cases, each SDN island and dynamic network among the islands, and data provisioning, processing and consuming equipments should be properly set up.

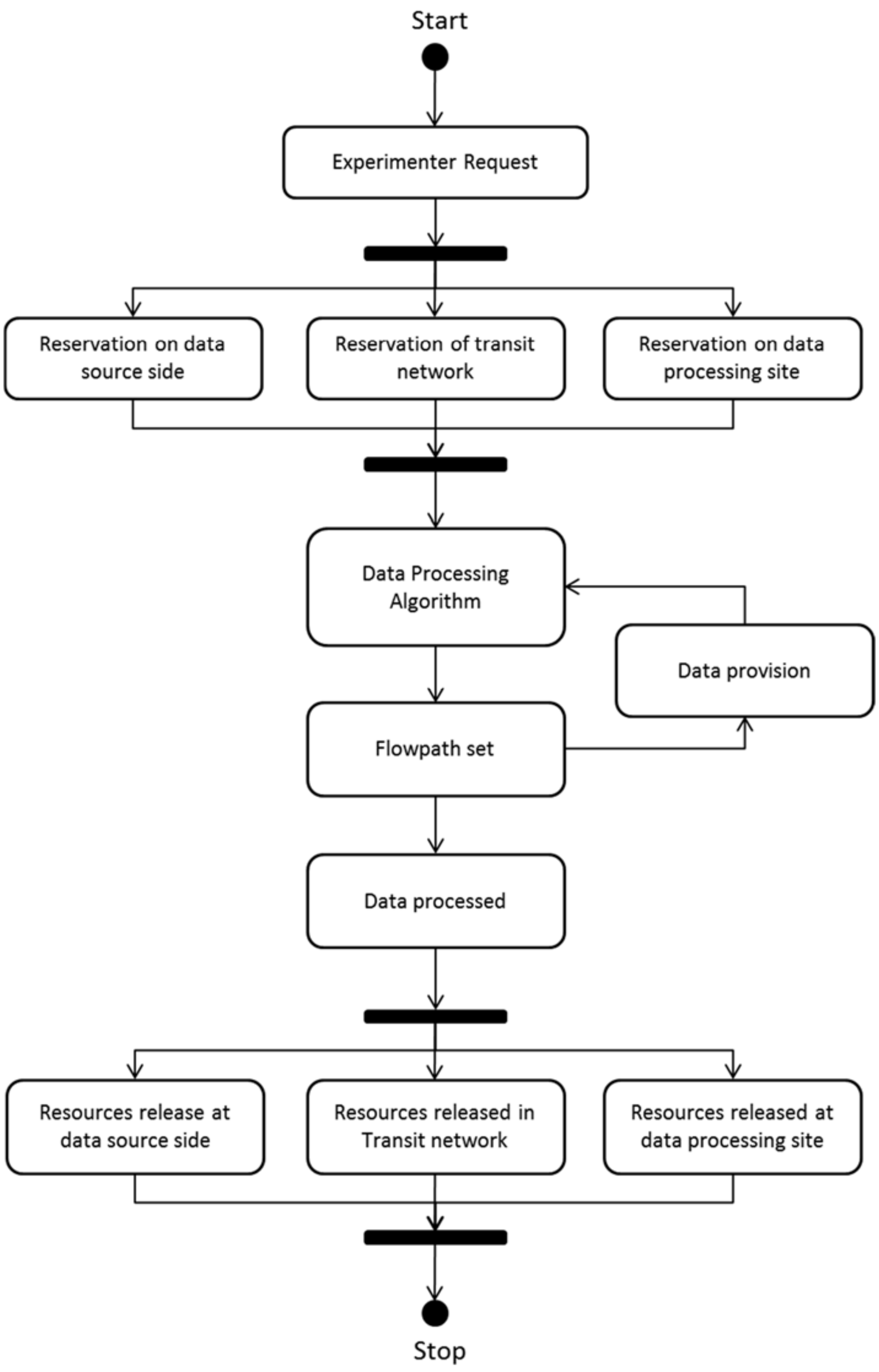
2.5.1 Data as a Service

Collaborative investigations generate large amounts of data that usually is stored in the place where it is generated. As an example, astronomical observations are performed around the world as it turns, and the data gathered in each site is stored locally. If all this data wants to be analysed, maybe it is not feasible or efficient to move it all to the place where it has to be analysed. A more efficient way would be to provide just the required data as it is needed to analyse. In order to maximize the efficiency, dedicated links from the data provider sites to the processing site could be enabled and disabled as the source of data changes. The combination of several SDN technologies like NSI and OpenFlow to establish dynamic links or flowpaths to provide data in a continuous and seamless flow is a key innovation to solve those kind of problems.

Use case	
Number	UC.1
Scenario / use case name	Data on Demand. Delivery of distributed data by setting data flows over the network.
Storyline	The problem exposed in this use case is how to process large amounts of data stored along different and distributed entities sites. We want to run an algorithm in a site called Data Processor over different portions of data that are located around the world in Data Storages. To do so, we need software that enables fast communication between different SaaS layers, providing the data as it is requested (Data on Demand). We also need to control the communication and well as establishing flows of data from the different Data Storages in an ordered way so data can be processed correctly by the user's algorithms at the Data Processor. In order to test such scenario we would need a distributed SDN-testbed such as the one that FELIX is proposing. FELIX testbed has control over the network elements, so it can define flow paths for the data in order to provide a channel between the Data Storages and the data processing site. This channel will be dynamically modified to serve the required data from the different data providers and receive a continuous flow of data with minimum delay between hops of sources. In the Data Processor, an application called Controller, has a global vision of a testbed network topology and can control the testbed network elements to define the flowpaths from sources to destination, while in the Data Storages, Data Agents send the required data through those flowpaths. When the experiment starts, the data processing algorithm requests for data to be provided. A controller knows where the data is allocated and sets a link between the two end points. Inside the data storage site, the data agent provides the data and, if necessary, defines a flowpath between the storage and the link point inside its network. When the controller knows that the required data is allocated in a different site, sets a new link between the data processing

	site end point and the new data storage site end point and disables the previous link. Links between end points are created and destroyed as requested until all the data is processed.
Goal (s)	The goal of this use case is to test the dynamic network reconfiguration by combination of NSI with SDN testbeds. The testbed will be able to set up the network to provide data from different sites in a continuous and seamless way to the processing site. From the Data Processing Site point of view, the data processed must look like a single repository providing data in a seamless way, keeping transparent the different locations of the Data Provider Sites.
Figure to visualize the use case/scenario	 FELIX testbeds Data Storage 1 Data Agent SDN Data Storage 2 Data Agent SDN . . . Data Storage N Data Agent SDN NETWORK NSI allows to define interconnection between testbeds. Data agents manage access to data in the Data Storage testbeds Control SW Data Processing Algorithm Data Center (FELIX Testbed)
Actors	Data providers Data Consumer (experimenter) SDN island operators Transit network operators
System components	**Several Data Storage sites** **A Data Processing site** **A Data Agent in each storage site** **NSI-controlled network domains** **SDN controlled network domains (L2)** **FELIX SDN Manager** **FELIX Resource Orchestrator** **FELIX Experiment Control and Management**
Preconditions	Several data providers with: • Data distributed in the different sites (no replicated data) • Data agents responsible to send the data when they are requested A data processing site with: • Capacity to run a data processing algorithm and send requests for data to the data providers A controller that receives requests from the algorithm and can define paths to connect the required data storage site with the data processing site.

Trigger	The experimenter starts running the data consumption algorithm.
Post conditions	The flowpaths required to deliver the requested data from the Data Providers sites to the Data Processing site are allocated and inter-connected as they are demanded. The data are effectively delivered in the most efficient way. Success conditions: • All data in the data providers testbeds must have been delivered to the Data Processing Algorithm • The data must be delivered seamless • The infrastructure configuration remains as it was before the experiment Failed End protection: • Dynamic network configuration should be reverted when experiment failed
Detailed Steps	

State Chart or Activity Diagram Explaining the steps in the scenario	 Start Experimenter Request Reservation on data source side Reservation of transit network Reservation on data processing site Data Processing Algorithm Data provision Flowpath set Data processed Resources release at data source side Resources released in Transit network Resources released at data processing site Stop
Explanation of "states" and "transitions" in state chart or "activities" and "flows" in activity diagram	There are three states identified for this use case: • Resource allocation: booking of resources to set the data processing machine and the data storage sites • Experiment execution: the algorithm requests data and the controller sets the flowpaths to the specific data storage sites • Resource release: after the experiment, the status of the testbed must be the same than before

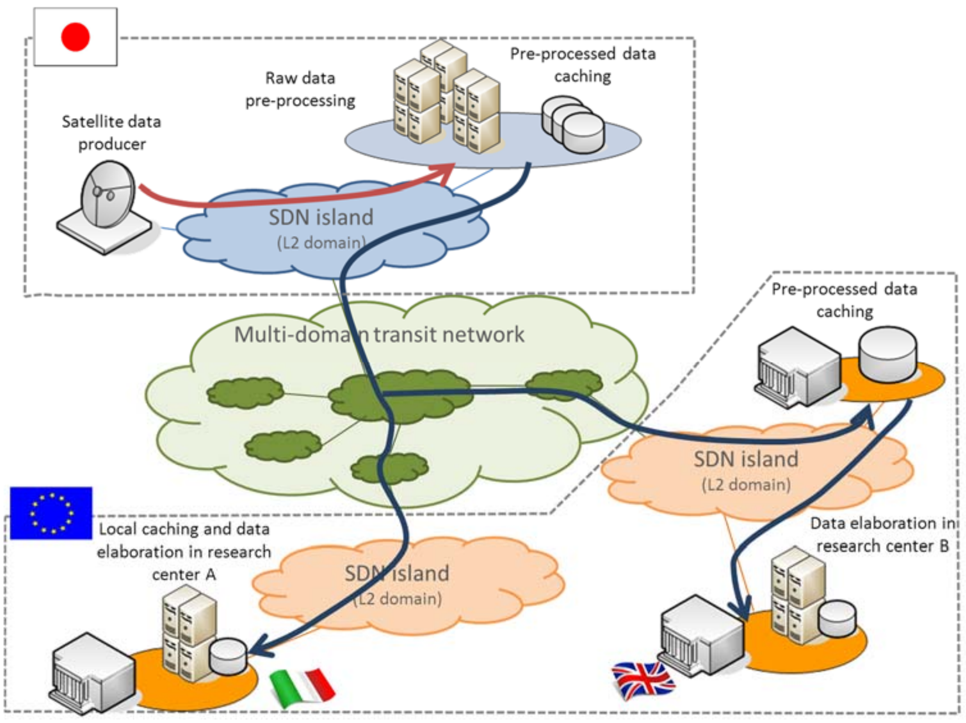
2.5.2 Data Processing Workflow

The delivery of nearly real-time data to geographically distant locations (i.e. from EU to JP and vice versa) is heavily impacted by the large Round Trip Delay (RTD) values achievable with transfers through the public Internet.

The FELIX system, based on the supervision/forecast of potential conditions for network degradation can allow to jointly allocate computing, caching and network resources and implement on-demand and application-driven network services for the specific data transfers. This is a key innovation point for FELIX, as it will overcome the currently best effort and limited interaction of remote communities of scientists.

This use case specializes the Data as a Service use case for the distribution of satellite images between researchers in different continents. Examples of source and consumer of satellite images are the ESA Earth portal (https://earth.esa.int/web/guest/home) and the JAXA Missions portal (http://www.jaxa.jp/projects/sat/index_e.html) and wide network of scientists in academia or research centers who use satellite data for their work.

Use case	
Number	UC.2
Scenario / use case name	Data pre-processing for minimizing network latency effect for live data
Storyline	The satellite sources generate huge amounts of nearly real-time data that can be pre-processed through computational functions placed close to the data producers. This approach reduces the size of data to be delivered across the transit network and elaborated at the geographically distributed research centers, improving the overall performance. The FELIX middleware can allocate the proper CPU and caching resources at data source and destination(s) ends, and configure the network segments on-demand with the required bandwidth and delay guarantees to provide an efficient transfer of data across continents.
Goal (s)	Major goal of this use case is to demonstrate and assess the capability of the FELIX system to provide a coordinated control of cloud and network resources. In this use-case, FELIX provides dynamic, on-demand and application-driven reconfiguration of end-to-end network connectivity across SDN islands inter-connected through an NSI-based multi-domain transit network.

Figure to visualize the use case/scenario	 Pre-processed data caching Raw data pre-processing Satellite data producer SDN island (L2 domain) Multi-domain transit network Pre-processed data caching SDN island (L2 domain) Local caching and data elaboration in research center A SDN island (L2 domain) Data elaboration in research center B
Actors	• A satellite data provider (in principle the providers of raw and pre-processed satellite data can be two different administrative entities; in this scenario we consider a single actor) • Researcher and experimenters (satellite data consumers) • Transit network operators (e.g. NRENs) • SDN island operators (e.g. NRENs) • Cloud providers (e.g. an IaaS provider)
System components	**SDN controlled network domains (L2)** **NSI-controlled network domains** **FELIX SDN Manager** **FELIX Resource Orchestrator** **FELIX Experiment Control and Management** **FELIX Monitoring Framework**
Preconditions	A satellite data provider with: • A running data acquisition system collecting satellite images and data • Algorithms for pre-processing of satellite information A research centre with • Algorithms for elaboration and analysis of pre-processed satellite data
Trigger	The research centre sends a request to start an experiment that requires the elaboration of the satellite information offered by the satellite data provider.
Post conditions	The IT resources needed to execute the experiment are allocated and inter-connected with the requested Quality of Service. The required satellite data are effectively received on the research centre at destination.

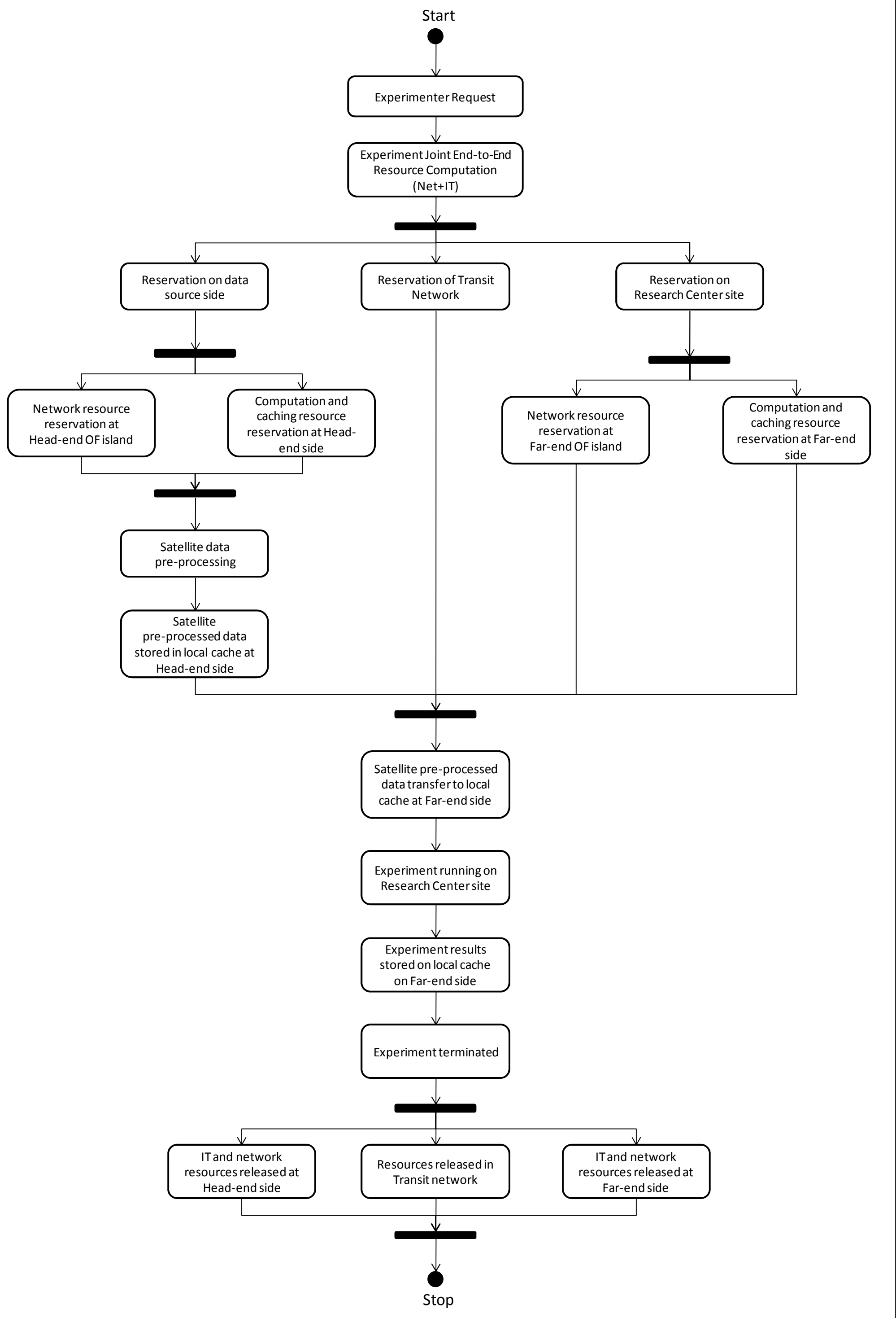
Success conditions:

- *Resource allocation on source end:* IT and network resources are successfully booked and provisioned on the data source side (e.g. Japan), close to the satellite source.
    - Computing resources are allocated on a data centre at the data source side to run the pre-processing algorithms for data preliminary elaboration and compression.
    - The network of the OpenFlow island on the source side is configured on-demand with dedicated connections from the satellite data source to the pre-processing site.
    - Pre-processed data is stored in local caches allocated close to the data source.
- *Resource allocation on transit network:* Resources within the transit network between the OpenFlow islands on the data source and destination research centre sides are booked and provisioned on-demand. Dedicated connectivity services are established to allow the efficient data transfer with the required RTD and throughput.
    - The resource provisioning in the transit network is mediated through NSI Agents called by the FELIX middleware.
    - Resource selection is performed using as objective function the trade-off between the achievable network performances and the application requirements.
- *Resource allocation on destination end:* IT and network resources are successfully booked and provisioned on the destination side (e.g. EU), close to the research centre where the data will be elaborated in the experiment.
    - Local caches are allocated within a data centre on the destination side to store the pre-processed satellite data delivered from the source end and to be distributed to the local experimenters' site.
    - Suitable computing resources are allocated to run the experiment on the research centre site.
    - The network on the OpenFlow island on the destination side is configured on-demand to enable the connectivity from the transit network to the local cache and from the local cache to the elaboration site.
- Experiment execution:
    - Pre-processed satellite data stored on caches in a data centre close to the final elaboration site (depending on the experiment requirements).
    - Final elaboration of pre-processed data running on VMs allocated on the research centre site.
    - Experiment results stored on caches in the research centre site.

Failed End protection:

- maintenance of consistency between the lifecycle of all the resources reserved for each specific experiment;
- in case of failure of a single resource, suitable recovery mechanisms must be applied depending on the configured policies; in case of failed recovery, all the resources related to the given experiment must be released.

Detailed Steps

State Chart or Activity Diagram Explaining the steps in the scenario	 Start Experimenter Request Experiment Joint End-to-End Resource Computation (Net+IT) Reservation on data source side Reservation of Transit Network Reservation on Research Center site Network resource reservation at Head-end OF island Computation and caching resource reservation at Head-end side Network resource reservation at Far-end OF island Computation and caching resource reservation at Far-end side Satellite data pre-processing Satellite pre-processed data stored in local cache at Head-end side Satellite pre-processed data transfer to local cache at Far-end side Experiment running on Research Center site Experiment results stored on local cache on Far-end side Experiment terminated IT and network resources released at Head-end side Resources released in Transit network IT and network resources released at Far-end side Stop
Explanation of "states" and "transitions" in state chart or "activities" and "flows" in activity diagram	Three major phases are identified for the use-case: • **Resource Reservation**, related to the specific booking phases on all the involved network and IT segments; • **Resource Commit**, related to the specific activation of the reserved resources (IT and Network) in all the segments at the proper time • **Experiment execution**, when the service is fully deployed and activated and users execute the experiment

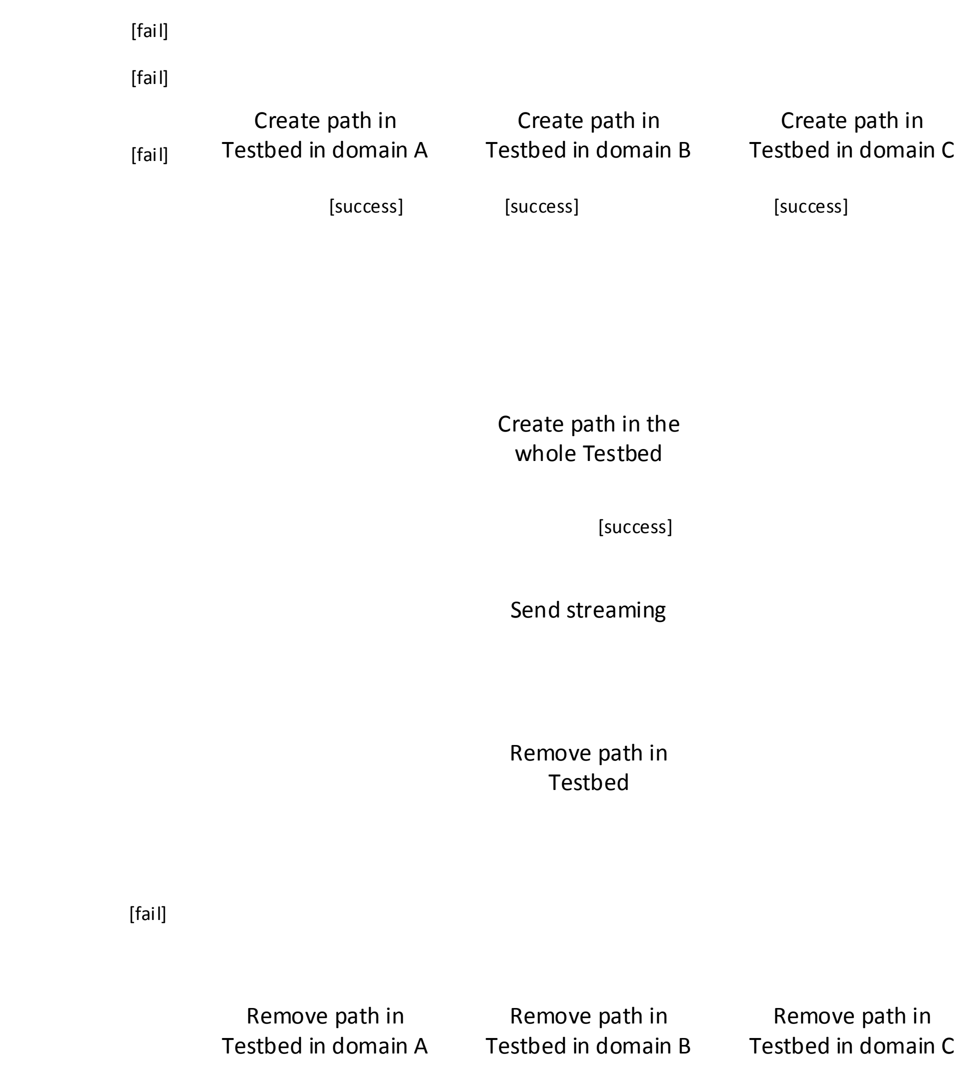
	• **Resource release**, when all the resources are de-commissioned upon experiment termination **These phases result in the following more detailed steps:** • **STEP 1 [Service request]:** The experimenter requests through the FELIX portal for the setup of IT resources and network connectivity required to allow the successful execution of his experiment, e.g. to elaborate information offered by the satellite data provider. • **STEP 2 [Planning / Design]**: The FELIX middleware verifies the availability of CPU and caching resources on the data source and destination research centre sides, in combination with the network capability to delivery traffic with the bandwidth, RTD and jitter constraints specified in the experiment description. The selection of the datacenters to reserve the IT resources can be performed taking into account the network condition (including historical data) and expected performance in the different segments (L2 SDN domains at source and destination, transit network). • **STEP 3 [Deployment]**: The FELIX middleware configures both the network and the IT resources selected in the previous step. The satellite data required by the experiment become available into the local caching close to the destination research centre (when required) and dedicated CPU and caching resources are ready to be used to execute the experiment and store the results. • **STEP 4 [Experiment execution and resource monitoring]:** the experimenter uses the set of resources made available for his experiments and launches his data processing workflow, consisting of raw data processing in CPUs and subsequent transfer and caching of the final products to remote sites. The status and performance of the established network services and reserved cloud resources is continuously monitored to verify the compliance with the description of the experiment requirements. No specific billing is foreseen for the execution of this use-case on the FELIX infrastructure. However, accounting needs to be properly activated on all the service segments (IT, network access, and network transit) in the perspective of any future sustainability of the infrastructure. • **STEP 5 [Decommissioning]**: When the experiment is concluded, all the related resources are decommissioned/released. Depending on local policies, cached data can be removed or maintained.

2.5.3 High Quality Media Transmission over long-distance networks

The evolution of media content delivery caused the ultra high definition video streaming to became quite common scenario, instead to be an exceptional case. The transmission of London Olympic Games 2012 from UK to Japan with 4K resolution was just a start, as Olympic Games in Tokyo in 2020 is already announced to be available in 8K resolution. Increasing quality is performed with the cost of network capacity, thus the high availability bandwidth pipes are became a requirement. But bandwidth in not a standalone attribute of the network. Researchers need to understand better the process of efficient distribution of ultra high definition content, how it influences the network, and how to optimize the hardware and its configuration for the best performance. The key point of this scenario is to understand how high demanding, large capacity traffic can be disturbed by network (L2 or SDN) and how such traffic can be optimized at global scale using EU-JP global testbed.

Use case	
Number	UC.3
Scenario / use case name	High Quality Media Transmission Over Long-Distance Networks
Storyline	Technology for automatic adjusting connection paths and parameters could act as powerful test engine for advanced multimedia purposes. High resolution visualisation solutions, especially streaming technologies, could be tested in order to pinpoint the sensitivity for network issues changes. Since they are very demanding and require high quality of transmission, therefore streaming can act as very sensitive test engines for transmission problems. While FELIX delivers a federation of SDN testbeds interconnected through NSI-enabled domains it enables opportunities for testing of media streaming over existing network technologies in backbone networks (DWDM, MPLS, etc.) in conjunction with OpenFlow deployed in the testbeds. In terms of these technologies the experiment would help to determine the behaviour of the transmission mechanisms in case of streaming high resolution multimedia content at a very long distance. The key aspect of such a kind of transmission is to take heed on sensitivity for negative effects in. There may be distinguish several aspects. First is jittering and lags, second wrong frames sequence, third are transmition disruption. Additionally when streamed media is 3D Video, then two streams have to be delivered separately for left and right eye. Synchronisation of these streams is extremely important from quality point of view and requires even more reliable transmission engines. The risk of the side effects may be reduced by transmission parameters, that would be under use case research. Therefore long distance network connections would be most factual test case for advanced content distribution systems. Extensive and properly planned tests could define possibility of future development and usage delivery systems regardless of the distance of transmission. Many aspects of this sensitive communication could be identified and qualified in order to define sufficient set of requirements for seamless streaming. Additionally, QoS as well as QoE (Quality of Experience) issues could be examined in order to find correlation between network problems and their visual reflection.
Goal (s)	The goal is to examine long distance network capabilities for media streaming. The network may be configurable in terms of flow and measured using various metrics. The software used for streaming may provide original solutions for side effects exclusion and quality measurement.

Figure to visualize the use case/scenario	Experimenter Domain A Domain B Domain C Media Consumers (End users) Testbed operator Media provider
Actors	Media provider, testbed operator, media consumer (end user), experimenter
System components	**FI experimental facilities** **NSI-controlled network domains** **Experiment Control and Management** **Monitoring Framework**
Preconditions	System must provide at least two machines connected through the federated testbeds over a long distance. One machine acts as a media provider, and second as a media consumer.
Trigger	Experimenter acting as a media provider user and media consumer user starts streaming manually
Post conditions	Success conditions: • Streaming is successfully provided by media provider and received by media consumer • Network Monitoring data are collected • Streaming quality is measured, side effects are addressed • Optionally – side effects are fixed either by network configuration or by streaming software solutions Failed End protection: • Dynamic network configuration should be reverted when experiment failed
Detailed Steps	

State Chart or Activity Diagram Explaining the steps in the scenario	 [fail] [fail] [fail] Create path in Testbed in domain A Create path in Testbed in domain B Create path in Testbed in domain C [success] [success] [success] Create path in the whole Testbed [success] Send streaming Remove path in Testbed [fail] Remove path in Testbed in domain A Remove path in Testbed in domain B Remove path in Testbed in domain C
Explanation of “states” and “transitions” in state chart or “activities” and “flows” in activity diagram	Activity (A), Flow (F) (A) **Create path in testbed in domain A (B, C)** – testbed operator creates path in different, independent domain (F) If every path in each domain is created correctly go to next activity otherwise finish operation

	(A) **Create path in the whole Testbed** – testbed operator creates the whole testbed to establish path in multidomain environment (F) If the whole path is ready go to next activity otherwise finish operation (A) **Send streaming** – experimenter send streaming from media provider to media consumer (F) When streaming is end, testbed operator remove all path in tesbed. (A) **Remove path in Testbed** – remove path in Testbed (F) When path between domains is removed go to next activity (A) **Remove path in Testbed in domain A (B, C)** – remove path in Testbed in each domain. (F) End operation

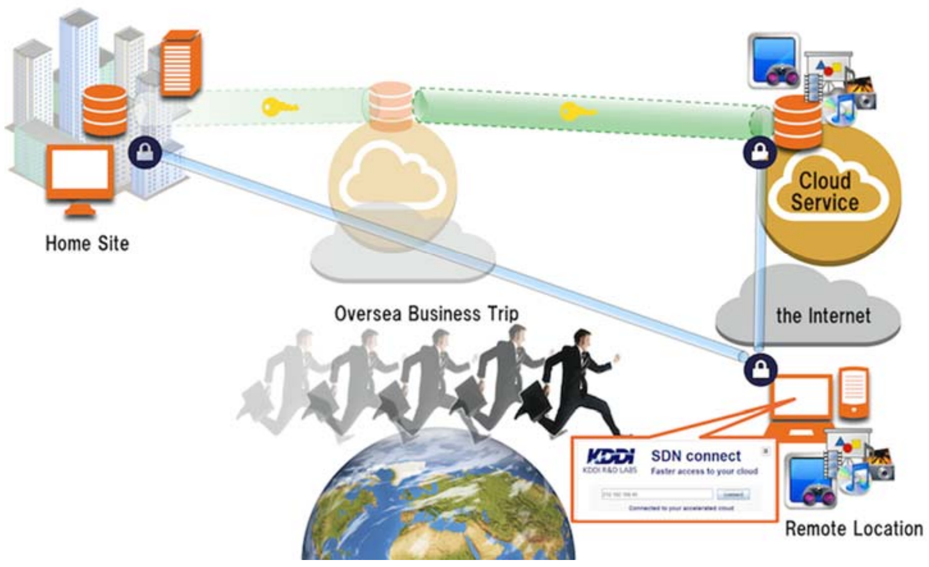
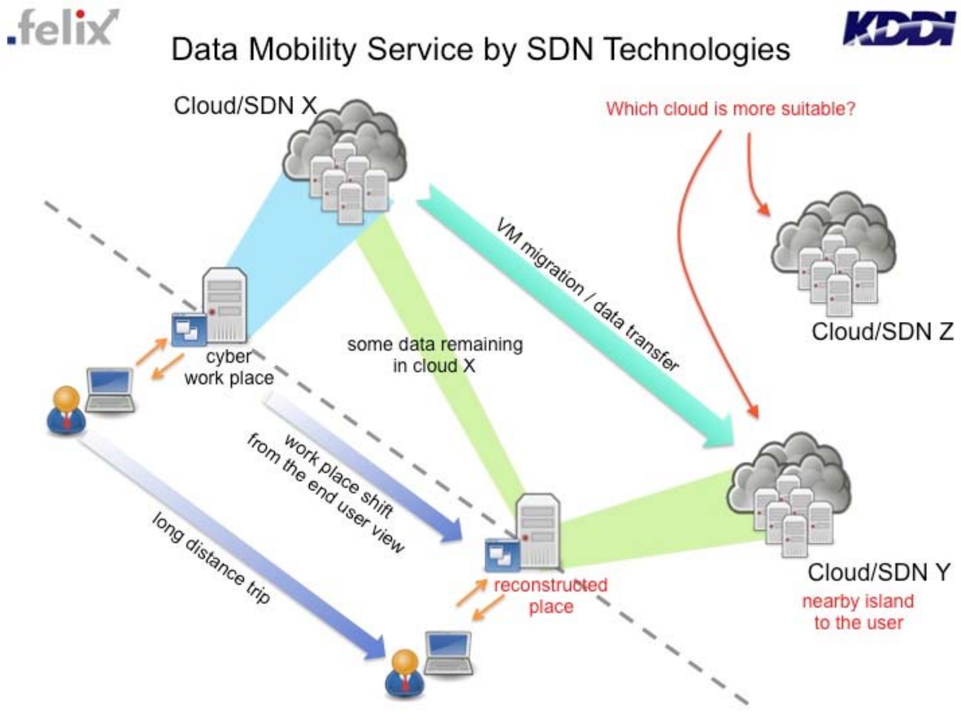
2.6 Infrastructure Domain Use Cases

Along the same lines as the Data Domain use cases, in the Infrastructure Domain use cases we also consider the same virtual infrastructure based upon federated resources. That is, the two domain cases do not differ in their architectural assumptions, or on their trust and security assumptions. On the contrary, we do plan to have a single architecture with enablers for both categories of use cases. The difference in the two domains is that in the Infrastructure Domain use cases, we do not focus on the efficient use of the network for data migration only but of entire workloads. Note that from the user's perspective there is nothing different to do to use the federation resources. This is all handled by the underlying mechanisms. The FELIX framework will provide a unified view on the infrastructure irrespective of the end user goals.
In this case, the virtual infrastructure consists of network, computing and storage resources which can migrate over the constructed SDN islands connected by dynamic circuit network. By dynamically configuring virtual infrastructures, flexible use of resources can be realized. Service can be provided by using physical resources which are convenient for the given purposes, such as better user experience, lower energy consumption and disaster resilience.

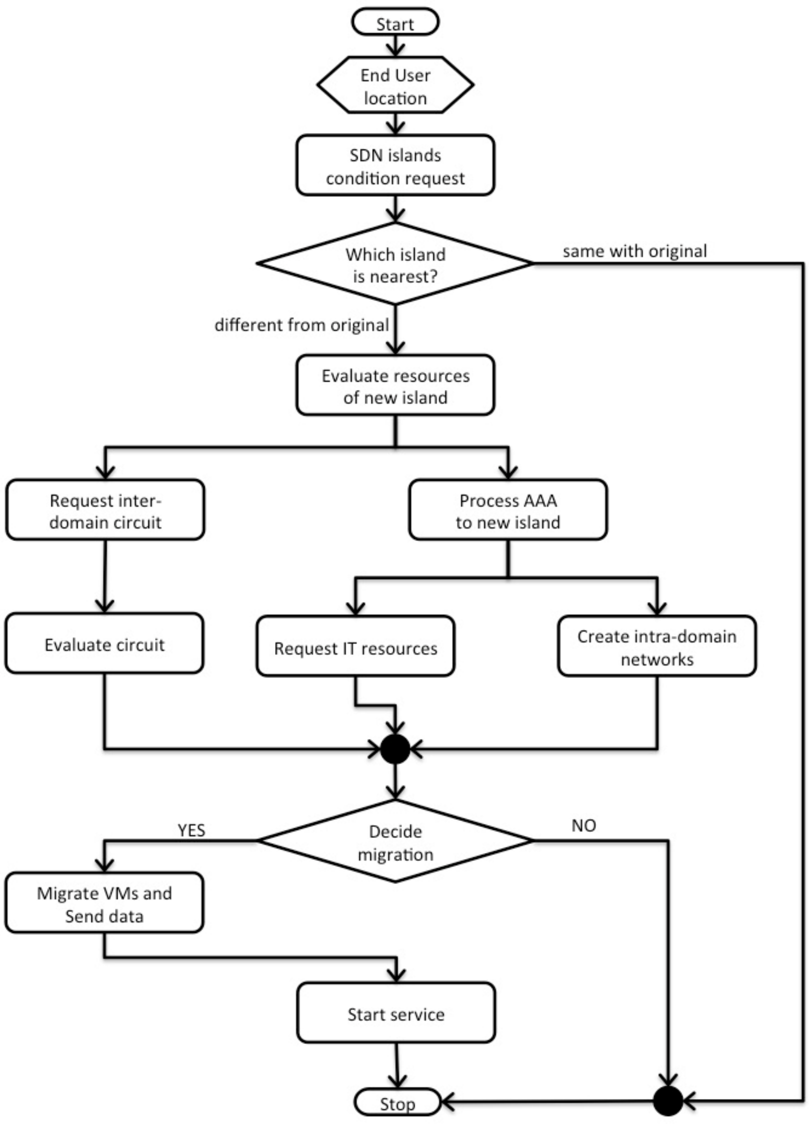
2.6.1 Data Mobility

In order to apply the cloud system to mission-critical areas, such as electronic administration, medical care, and finance, it will be essential to meet demands for end-to-end guaranteed service quality, reliability of compliance, and power saving of cloud system. However, it is difficult to continue to provide a stable service by single-cloud system because there is a limit to available reserve resources. In order to guarantee the required service quality, such as service availability and performance, even in such cases, there will need to provide a mechanism for flexibly reassigning resources among cloud systems by means of an inter-cloud system. Especially, since private clouds are designed and built with full capacity using virtualization technology, it is difficult to provide prompt services according to the needs of end-users (e.g. high-performance, location, policy). Therefore, it is even more important in such cases to enable a cloud system to work with other cloud systems on a temporary basis. While at the same time, software-defined network (SDN) is expected to become major network technology for cloud system in a data-center. So when actually applying inter-cloud to cloud systems on the market, it is important to satisfy demands for guaranteed end-to-end service quality among SDN enabled cloud networking across multiple data-center.

Use case	
Number	UC.4
Scenario / use case name	Data Mobility Service by SDN Technologies (Inter-Cloud use case)
Storyline	In order to apply the cloud system to mission-critical areas, such as electronic administration, medical care, and finance, it will be essential to meet demands for end-to-end guaranteed service quality, reliability of compliance, and power saving of cloud system. However, it is difficult to continue to provide a stable service by single-cloud system because there is a limit to available reserve resources. In order to guarantee the required service quality, such as service availability and performance, even in such cases, there will need to provide a mechanism for flexibly reassigning resources among cloud systems by means of an inter-cloud system. Especially, since private clouds are designed and built with full capacity using virtualization technology, it is difficult to provide prompt services according to the needs of end-users (e.g. high-performance, location, policy). Therefore, it is even more important in such cases to enable a cloud system to work with other cloud systems on a temporary basis. While at the same time, software-defined network (SDN) is expected to become major network technology for cloud system in a data-center. So when actually applying inter-cloud to cloud systems on the market, it is important to satisfy demands for guaranteed end-to-end service quality among SDN enabled cloud networking across multiple data-center. A user of a service provided by a cloud system moves to a remote location (e.g., on a business trip). Because of the longer physical distance (causing a longer network delay) from the home site where the service is provided, the user experiences degradation in response performance. The cloud system detects the degradation in response performance. In order to provide the same service in similar performance as before at the remote location, it leases some resources from a cloud system located close to the user visiting place, and transfers user's ID, application, and data to this cloud system so that the user can access the service provided from a site near his or her current location. As a result, the user can temporally access the service with the same ID and at the same level of performance as before. Use case failure means user can't get the same level of network performance as before.
Goal (s)	The goal is to verify the end-to-end tenant and network connectivity across SDN testbeds that are created by the interworking of cloud agent's SDN controller and FELIX management system. The user's data (user's ID information, application) are temporarily move to the cloud system near a user's location among SDN islands interconnected by NSI based transport network.
Figure to visualize the use case/scenario	

	1. Concept picture of inter-cloud service  Home Site Cloud Service Oversea Business Trip the Internet SDN connect Faster access to your cloud Remote Location 2. Bacic design of experiment  Data Mobility Service by SDN Technologies Cloud/SDN X Which cloud is more suitable? VM migration / data transfer Cloud/SDN Z cyber work place some data remaining in cloud X work place shift from the end user view long distance trip reconstructed place Cloud/SDN Y nearby island to the user
Actors	< the individuals and their means that trigger system reactions > Inter-cloud service agent : I-CSA Users of inter-cloud agent service Physical cloud and network provider alliance in JP and EU
System components	API between experimenter's SDN controller **FI experimental facilities** **NSI-controlled network domains**

	SFA-based federation **SDN Manager** **Resource Management** **Experiment Control and Management** **Monitoring Framework**
Preconditions	Inter-cloud user: • Access the cloud system at usual locations I-CSA : inter-cloud service agent: • Manage the resource and availability of distributed cloud system • Manage the user location based on user account ID • Manage the which cloud system is being used for each user Physical cloud and network provider: • Run the SDN based tenant network for Inter-cloud user • Monitoring the SDN based tenant network to provide the data with I-CSA
Trigger	The user reconnects the usual home cloud system remotely via VPN connection
Post conditions	< those conditions that could occur once the use case has been executed and the system continues its operations > It should explain when the execution of a use case scenario can be considered as a success or failure. The section should be short and not extremely technical. **Success conditions:** The cloud system detects the degradation in response performance. In order to provide the same service in similar performance as before at the remote location, it leases some resources from a cloud system located close to the user visiting place, and transfers user's ID, application, and data to this cloud system so that the user can access the service provided from a site near his or her current location. As a result, the user can temporally access the service with the same ID and at the same level of performance as before. Failed End protection: • When the experiment failed and operation time over the defined timeout period, all the dynamic configuration include other domain should be removed and revert to the previous state. • The re-request by user mechanism for user, when the data does not moved to the optimal cloud
Detailed Steps	

State Chart or Activity Diagram Explaining the steps in the scenario	 Start End User location SDN islands condition request Which island is nearest? same with original different from original Evaluate resources of new island Request inter-domain circuit Process AAA to new island Evaluate circuit Request IT resources Create intra-domain networks Decide migration YES NO Migrate VMs and Send data Start service Stop
Explanation of "states" and "transitions" in state chart or "activities" and "flows" in activity diagram	Three major steps and states are identified for the use-case: **A: Decision of End user Location** : I-CSA determine the user's location by the notification function between I-CSA and user. **A: SDN islands connection request** : I-CSA request the resource condition and select the most suitable resource. **F: same with original** : If selected resource is same with home site, the experiment is over. **F: different from original** : If selected resource is different from original go to next activity. **A: Evaluate resources of new island :** I-CSA evaluate the resources by the Information of IT condition and monitoring data which provide by IT FELIX management system. In this phase I-CSA start two function. One is sending the request of Inter-domain circuit between home-site to new SDN island, and the other is process AAA for IT resource in new island. **F: Requesting Inter-domain circuit :** I-CSA request the Inter-domain circuit via NSI and inter-domain circuit set up in this phase.

	F: Requesting intra-domain netowork : I-CSA request the intra-domain network to new SDN island through the own SDN controller function. **A: Decide migration :** Commit and activate the whole resources and intra/inter-domain network connectivity between home and remote SDN island over multi-domain environment. If I-CSA confirmed the all resources, go to migration phase. If all resources not activate, report the error to user and experiment stop and dismiss. **A: Data migration** : I-CSA starts the migrate VM and data from home cloud system and remote cloud. **A: Start service** : I-CSA report the activation of all resource to user and service start **A: Resource release and stop** : All the resources are released when user request deactivate the ervice or when user retune the home-site.

2.6.2 Follow the [Moon|Sun]

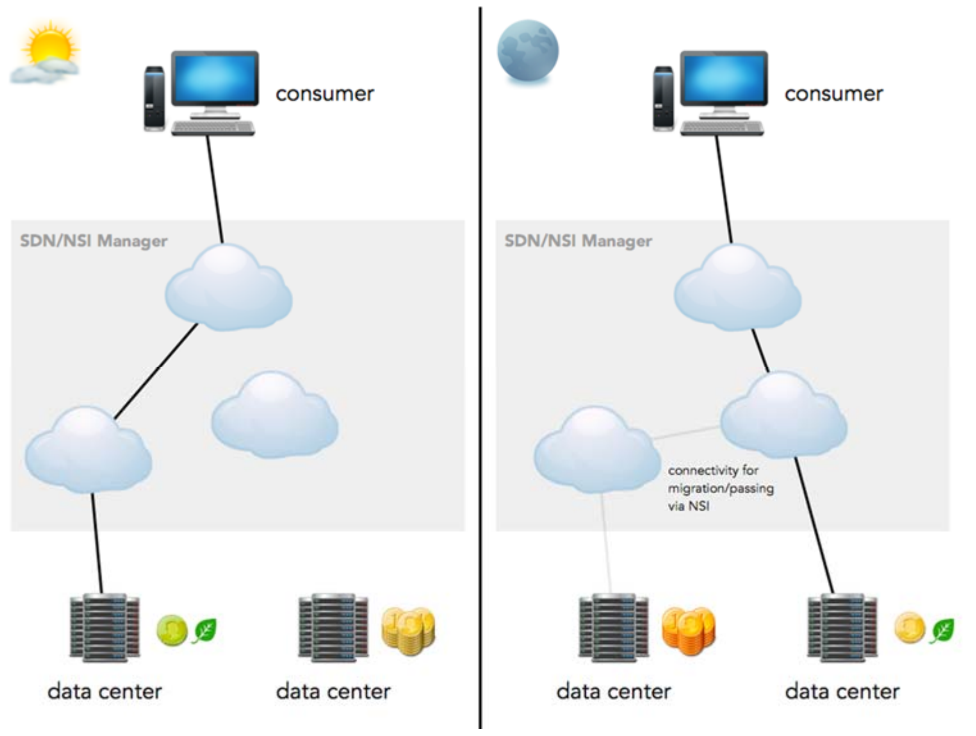
This use case motivates the need for dynamic provisioning of networking resources for high-performance computing (HPC) across a range of federated and pooled network, computing and storage resources. One particular application of the enablers that are necessary for implementing such a use case is modulating the energy consumed by the federated infrastructure. For example, renewable energy sources do have a better carbon footprint for powering the federated infrastructure but their supply cannot be controlled with the same precision as with fossil fuel power plants. By enabling workload migration between data centres, users of the federated infrastructure, for instance, will be able to automatically use the most environmentally-friendly and (energy) efficient data centre available.

This use case introduces a several features beyond the current state of the art in world-wide testbed federations:

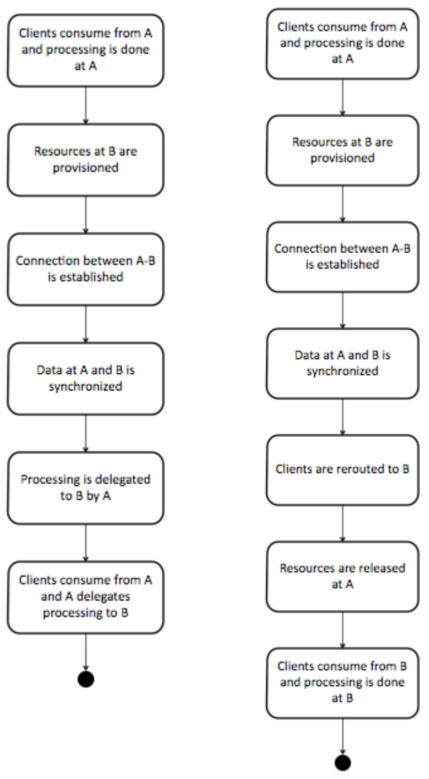
- Establishing inter-provider connections on demand
- Dynamically provisioning compute nodes load
- Synchronizing (consumer) data
- Transparently re-routing consumer traffic
- Delegating processing algorithmic and subsequent proxying
- Establishing trust: Providing a scheme to perform authentication and authorization for management actions

Use case	
Number	UC.5
Scenario / use case name	Follow-the-sun / follow-the-moon principles
Storyline	The original idea which served as inspiration for this use case is described in detail in [7]. Therein, the authors state that Internet usage curves are following a similar daily pattern everywhere in the world, and that there is a natural potential to shift the load of data centres to places in the world where it is currently night (“follow the sun/moon”). Moreover, the prices of renewable energy strongly depend on the availability of wind and solar energy. As cooling is reported to amount up to

	50% of the total energy bill of data centres, recently, large data centre providers have started to place infrastructure in formerly unusual places such as Iceland and Finland. There, the data centres can leverage the low environmental temperatures for cooling, for example. Similar projects may be considered in desert areas, where the amount of solar energy available may even make up for the additional cooling required. In order to leverage these monetary and environmental effects, the load of one data centre needs to shift to another. This shift can be done following two approaches. First, the workload is moved to the more efficient data centre entirely and the consumers' traffic is rerouted. Alternatively, a less drastic option is to handle the consumers' requests at the less efficient centre and delegate the actual processing to the more efficient data centre. This way, the hosts in the first data centre can act as proxies and workload load balancers outsourcing part of their compute/storage/networking effort. These kind of proxies naturally have a lower resource consumption, so the less efficient centre can reduce its load. The second option requires the service provided by the data centre to enable delegation. This means that the algorithm for processing must enable distributing the computation load. Both scenarios - complete migration and delegation – have a strong need for establishing dynamic, on-demand end-to-end connections between the data centres, which can be seen and implemented in practice as federations. NSI is a technology which can fulfil this requirement. Another strong requirement is the use of SDN mechanisms for (re-)routing of the consumers' traffic, the traffic within the data centres and between them. When the workload is moved from one data centre to another, compute resources need to be provisioned, as discussed above. The more efficient data centre gets notice to allocate more resources and once they are available the traffic migration can be triggered.
Goal (s)	The overall goal is to minimize the total cost of operation for a virtual data centre while maintaining an acceptable end-user experience. This cost is mainly determined by the energy costs, but may also include cost for CPU/RAM usage and bandwidth cost. Another goal is to regulate the usage of resources in a data centre. This may be used to shape the consumption according to the availability (time and place) of renewable energy sources, for example. In terms of the goals for the FELIX project, this use case can demonstrate and assess the capability of the system in the following ways: • Provision of end-to-end network connectivity via an on-demand NSI-based multi-domain transit network. • Usage of SDN capabilities to provide/reconfigure the routes within the SDN islands (data centres). • Dynamic allocation of (compute) resources as a pre-condition to shift load.

Figure to visualize the use case/scenario	 consumer SDN/NSI Manager data center data center consumer SDN/NSI Manager connectivity for migration/passing via NSI data center data center This figure depicts the scenario, re-routing all traffic to the more efficient data centre.
Actors	Data consumer Two or more data centres/SDN islands (with different characteristics/locale, e.g. using energy with different price functions, be located in cooler regions in the summer time and so on) Transit network operators
System components	**FI experimental facilities** **NSI-controlled network domains** **SDN Manager** **SFA-based federationResource Management** **Experiment Control and Management** **Monitoring Framework**
Preconditions	At least two data centres must be available. In the case of full migration, both data centres must be able to host the consumers' services/data. In the case of only delegating the processing to the more efficient data centre, the more efficient data centre must be able to host the delegated load (in terms of capacity). Both data centres must be able to dynamically provision compute resources (or have them available permanently). Both data centres trust each other (in terms of authentication and authorization). The data consumer uses resources/data provided by one data centre. The consumer acquires the resources/data from the data centre with the cheaper operation cost.

Trigger	We could envision several triggers, but one example is that the operation cost (energy cost) of the data centre used by consumers rises above the estimated cost of handling the costumers in another data centre.
Post conditions	Success conditions: The consumers' data/processing load was successfully migrated from one data centre to another. The operational cost of the chosen data centre is below the previously chosen data centre. In the complete migration scenario: The traffic of the consumer is routed to the newly chosen data centre. In the delegation scenario: The processing load is shifted to the newly chosen data centre. The consumer did not experience synchronization problems. Ideally the migration process is transparent to the user. The system established new inter-provider connections via NSI if necessary (for the migration and/or the customer connectivity). Failed End protection: The consumers' data needs to be rolled back to a consistent state. The consumers shall be connected to the data centre used before the failing. (NSI) connections established during the migration need to be reverted.
Detailed Steps	

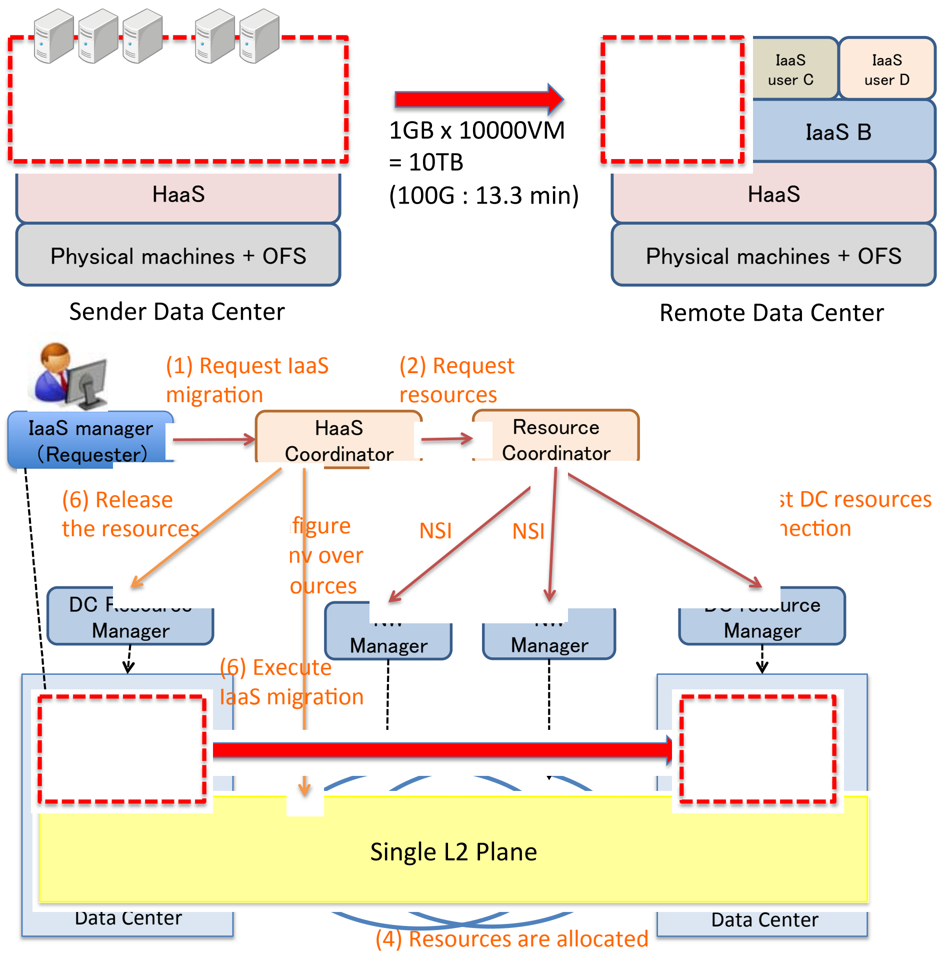
State Chart or Activity Diagram Explaining the steps in the scenario	 Clients consume from A and processing is done at A Resources at B are provisioned Connection between A-B is established Data at A and B is synchronized Processing is delegated to B by A Clients consume from A and A delegates processing to B Clients consume from A and processing is done at A Resources at B are provisioned Connection between A-B is established Data at A and B is synchronized Clients are rerouted to B Resources are released at A Clients consume from B and processing is done at B Left: Full migration scenario. Right: Delegation scenario
Explanation of "states" and "transitions" in state chart or "activities" and "flows" in activity diagram	A and B stand for "data centre A" and "data centre B". The first state is the initial situation for the use case. The change to the second state is triggered when the operation cost for B is expected to be lower than the cost for A. The goal is achieved when the last state is reached and the load at A can be reduced. Provisioning resources at B creates new computing resources at the more efficient data centre. This step ensures that the additional load introduced by the migration/delegation can be handled. Establishing the connection between A and B requires the usage of NSI to establish end-to-end, inter-provider connectivity. The synchronization of data between A and B is optional and may not be required depending on the consumed services. Scenario on the left: The rerouting of the consumer may require establishing connectivity via NSI and it is likely that SDN concepts are needed to transition the users' traffic from A to B.

	Scenario on the right: When A starts delegating the load to B, the hosts at A become proxies. Hence, the work load becomes less and the energy consumption drops.

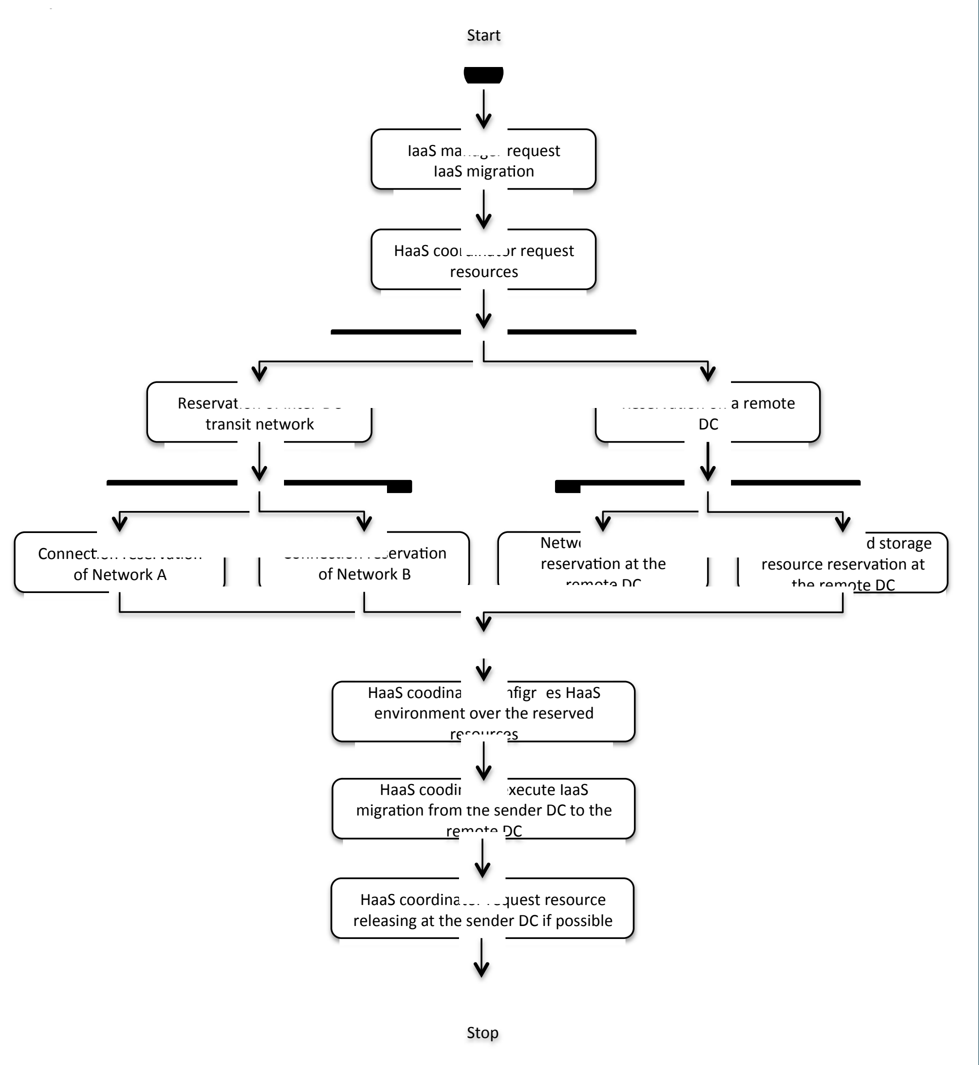
2.6.3 Disaster Recovery by Migrating IaaS to a Remote Data Center

Business Continuity Planning (BCP) of IT services is one of important issues for Cloud users and providers especially after the great east Japan earthquake in 2011. Typical Cloud services are managed by an Infrastructure as a Service (IaaS) software, such as OpenStack and CloudStack, and provide isolated tenants on physical resources, such as computers, storage and network, in a data center with multiple IaaS users. Howerver, it is difficult to continue to provide such IaaS services when a serious disaster happens because electric power supply will be stopped.
The FELIX middleware will enable migration of whole such an IaaS, which consists of a cluster of virtual machines including IaaS management node, to a remote data center to continue each business.

Use case	
Number	UC.6
Scenario / use case name	Disaster recovery by migrating IaaS to a remote data center
Storyline	Infrastructure as a Service (IaaS) providers prepare physical resources, such as computers, storage and network in a data center, and install IaaS management software, such as OpenStack and CloudStack, on the resources. Then IaaS providers provide isolated tenants, which consist of virtual machines, isolated storage volumes and isolated network, with multiple users, who create and provide business or entertainment application services on the provided tenants, respectively. Users of such IaaS expect the IaaS environment to be stable and fault free. However, It is difficult to continue to provide such IaaS services when a serious disaster, e.g., the great east Japan earthquake, happens because electric power supply will be stopped. In order to continue a variety of services on an IaaS, all of IaaS resources have to be migrated to another site, which is not damaged, while UPS and equipped simple power generator are available. This use case will realize such IaaS migration between data centers in Europe and Japan. Hardware as a Service (HaaS) is a key idea to realize this IaaS migration. HaaS is a service, which dynamically configures and provides virtual resources, on which IaaS can run, by using nested virtualization technologies, such as KVM and OpenFlow. The virtual resources running on the HaaS layer of a data center can be migrated on the HaaS layer of another data center. In addition, high bandwidth internetworking technology will enable this IaaS migration performed in a limited duration.
Goal (s)	Migration of a working IaaS on which multiple tenants are being executed from a data center to another data center.
Figure to visualize the use case/scenario	

	 HaaS Physical machines + OFS Sender Data Center 1GB x 10000VM = 10TB (100G : 13.3 min) IaaS user C IaaS user D IaaS B HaaS Physical machines + OFS Remote Data Center (1) Request IaaS migration (2) Request resources IaaS manager (Requester) HaaS Coordinator Resource Coordinator (6) Release the resources figure nw over ources NSI NSI st DC resources nection DC Resource Manager Manager Manager DC resource Manager (6) Execute IaaS migration Single L2 Plane Data Center Data Center (4) Resources are allocated
Actors	An IaaS sender, which runs an IaaS software (e.g., CloudStack) on a HaaS layer at a data center. An IaaS receiver, namely an infrastructure provider, which provides HaaS in a data center. Network, which provide a dynamic network circuit between the data centers from and to which an IaaS is migrated.
System components	**FI experimental facilities** Sender side: a data center running an IaaS software including a manager, on HaaS. Receiver side: a data center, which enables to provide HaaS. **NSI-controlled network domains** **Inter-data center network provider software** **SFA-based federation** **Inter-data center resource management software** **SDN Manager** **Resource Management** **Data center resource management software including SDN managers and computer and storage resource managers** **Experiment Control and Management** **HaaS coordination software**
Preconditions	MUST

	• Two HaaS-enabled data centers • IaaS running at the sender data center. • The destination data center has enough physical resources. • The destination data center can emulate a physical resource environment of the migrated IaaS. -> Resources of the destination data center support nested virtualization technologies. DESIREBLE • The network bandwidth between the source and destination data centers can be assured. • Software acceleration, such as High performance data transfer under a high latency environment, and high speed VM migration technologies, is available.
Trigger	Emulated disaster event.
Post conditions	**Success conditions:** <which conditions demonstrate the correct and successful completion of all the steps in use case> MUST • A working IaaS and tenants on the IaaS are migrated from a data center to another data center, according to a trigger event. • Tenants on the migrated IaaS continue operation. ADVANCED • IaaS image is cached at the destination data center and only updated portion of the image is transferred at the time of migration, so as to realize quick migration of processing. **Failed End protection:** It will be failed if all of the above MUST conditions are not satisfied.
Detailed Steps	

State Chart or Activity Diagram Explaining the steps in the scenario	
Explanation of “states” and “transitions” in state chart or “activities” and “flows” in activity diagram	At a sender side, an IaaS is run on top of a HaaS layer over the physical resources. Multiple tenants will be run on the IaaS. An IaaS migration process start when a trigger (disaster) happens at the sender side data center. 1. The IaaS manager requests migrating his IaaS to the HaaS coordinator. 2. The HaaS coordinator requests required resources to a resource coordinator. 3. The resource coordinator co-allocates resources to a remote data center resource manager, and network connection between the sender and the remote data centers to a network resource management system. 4. The HaaS coordinator configures a HaaS layer over the allocated network and data center resources. 5. The resource coordinator executes IaaS migration from the sender data center to the receiver data center. 6. The IaaS continues to run at the receiver data center. The resource coordinator releases the resources of the sender data center if possible.

Start
IaaS ma request IaaS migration
HaaS coo r request resources
Reservat transit network
a remote DC
Connect of Network A
ervation of Network B
Netw reservation at the
d storage resource reservation at the remote DC
HaaS coodina nfigr es HaaS environment over the reserved
HaaS coodi xecute IaaS migration from the sender DC to the
HaaS coordina quest resource releasing at the sender DC if possible
Stop

3 FELIX User Scenarios Requirements

The use cases specified in Chapter 2 originate a set of user requirements that mostly relate to the expectations on the FELIX system in terms of objectives, use-case environment, constraints, and measures of effectiveness and suitability. These requirements emerged during the use cases identification work and represent an initial input for the FELIX architecture. At this stage, they cannot be considered consolidated architecture requirements of the FELIX framework, and will be further refined, groups and prioritized during the execution of the project to better reflect the functionalities implemented by the system.

The requirements presented in this section are grouped in three sets, depending on the assessment made by the FELIX partners on the importance of the particular requirement for the implementation of a use case. This importance ranking is expressed according to IETF RFC 2119 with the following keywords:

- **MUST** – for requirements which are mandatory for design and implementation of the FELIX framework and its components,
- **SHOULD** – for requirements which are recommended due to increased efficiency, optimization or any other positive effect to the end users
- **MAY** – for requirements which are optional and do not influence overall FELIX functionality in significant manner.

It is worth to note that the FELIX requirements defined in the document are not final, and will be further refined and consolidated while the overall FELIX architecture is being defined. This list will be investigated in details, prioritized and translated into FELIX framework requirement list in deliverable D2.2 (FELIX Framework Architecture), in which technical references to specific FELIX system components will be also identified.

ID	Requirement	Description
UR.1	The FELIX orchestrator MUST be able to allocate computing and storage resources in the different remote SDN islands	The FELIX orchestrator MUST be able to request, synchronize, schedule, verify, and tear down allocation of computing and storage resources in the different island of federation and within a specific slice. It is not decided yet, whether the orchestrator has a role of a super SDN controller, or provides just an interaction and synchronization mechanism for SDN controllers in particular federations.
UR.2	The orchestrator MUST be able to request a link service	The orchestrator MUST be able to request a connection service over the transit network between distant federated islands, using

	(connection) over the transit networks through the NSI and/or SDN interfaces/API	the NSI interface and proper flow processing in the L2 SDN-based domains. The network between the federated islands can be delivered by non-SDN domains, with heterogeneous environment. The assumption was made that these domains MUST be dynamically managed, due to FELIX behavior. Any management/control system can be used for this purpose, however such system MUST support NSI v2.0 protocol as an interface for accepting provisioning request. The SDN based networks must deliver an API/interfaces for the orchestrator, in order to extend the long distant NSI connections to the computing/storage resources within slices.
UR.3	The FELIX framework MUST allow the dynamic addition, removal and modification of network flows	The FELIX framework MUST operate in dynamic environment, getting and releasing the resources as needed. All actions taken by the FELIX orchestrator must be a result of a calculations performed by some kind a scheduler, which takes information on resources availability and reachability as an input. There are no static configuration of connectivity between the SDN islands, not within those islands, and there are no static slices configuration. The FELIX framework must be supported by NSI networks, SDN controllers and any other controlling entities in his decisions and configuration implementation. The resources may added, removed, or possibly modified at any moment, with instant effect to all controlled infrastructure. The resources MUST NOT be allocated beyond duration of particular experiment.
UR.4	Proper algorithms for automatic flowpath allocation MUST be available for use	FELIX must provide algorithms allowing selection of resources (including network) for particular requested slice. User should not be concerned about any resources selection, however he/she SHOULD have an influence on selection of resources, if Use Case require so (e.g. selection of particular SDN island to use or usage of particular long range links).
UR.5	Proper authentication and authorization mechanisms MUST be available to allow access to the different data storage sites	Security is one of the most important features, required for creation of operational services in the future. Therefore all FELIX components which are involved in communication through a network should implement basic authentication and authorization mechanisms (e.g. ciphering with SSH protocol, usage of signed certificates for authentication, etc.).
UR.6	FELIX MUST create a circuit via multiple domains via NSI protocol	Since FELIX framework is using an NSI protocol it must be able to operate in multi-domain environment. The FELIX test-bed will be an example of such environment, where connections from EU to JP will be implemented in cooperation of several domains, at least involveing NetherLight/SurfNET and JGN-X.
UR.7	FELIX MUST deliver monitoring of the resources behavior	FELIX framework must provide at least basic monitoring capabilities of the network and IT resources behavior and efficiency. This includes as an example: • Interface availability (e.g. via ping) • CPU/Mem/HD usage/availability on a host (physical) • Network RTT, Jitter, Throughput (if possible)
UR.8	FELIX MUST notify requestor on slide events	FELIX framework needs to interact with end users, by notifying them on important events. This includes at least slice resources

		delivery and tear down. It would be also useful to notify users about failures, encountered efficiency problems, and taken automatic repair actions by the framework.
UR.9	FELIX MUST have a Flexible VPN client software which inter-work with FELIX management system	Access to remote resources in FELIX slices will require a VPN client, which will ensure proper security. The VPN system must compatible with FELIX management system, allowing automated configuration and providing desired level of accessibility of resources from public Internet.
UR.10	FELIX MUST provide APIs for inter-work among FELIX management system and cloud agent SDN controller	FELIX will require to define an API for integration of NSI and SDN environments, so that information about resources request can flow in a controlled manner. The integration of the two words is the main goal of the FELIX, thus this requirement is critical for FELIX to operate.

Table 2: MUST FELIX users' requirements set

ID	Requirement	Description
UR.11	FELIX orchestrator SHOULD benefit of a single path computation entity and service orchestration tool for joint selection of network and IT resources depending on the experiment requirements and the capability and resource availability within the data centres on the far-end side	Instead of using two logical entities, one for setting up SDN slices, and the other for connections between SDN islands, the FELIX orchestrator could unify these features into one functional block. The result would be an opportunity to optimize the path and resources allocation using various constraints, e.g. distance, efficiency, resources availability, energy saving factors, etc. This feature would possibly allow for better resources allocation and improved scheduling of resources. However, this functionality is expected to be quite complex at this moment and is not vital to FELIX framework to operate.
UR.12	FELIX orchestrator SHOULD benefit of an embedded NSI Aggregator function to effectively interact with a multi-domain transit network.	The NSI Aggregator is a feature allowing to know global connection schema of all NSI enabled network. This knowledge allows the NSI agents to perform i.e. path finding functionality and take decisions on how to route particular circuit. It also allows to create a hierarchical model of NSI network dependencies, as multiple NSI network can be managed by single NSI Aggregator. The can be more than one Aggregator in the overall network, making the infrastructure more scalable and facilitating its controlling. The orchestrator could include NSI Aggregator feature into its architecture, making it aware of global network connectivity and allowing for path computation. Also enhanced path optimization can be performed in such case. Having an NSI Aggregation feature inside the orchestrator is not critical however, as an external NSI agent can provide such functionality. This will obviously reduce the control over the NSI environment by orchestrator, but the FELIX framework will still be able to operate normally.
UR.13	The FELIX framework SHOULD benefit of tools for collecting continuous monitoring information (e.g. perfSONAR)	The monitoring information may be used to take enhanced decisions about network and IT resource allocation or re-configuration, e.g. for service recovery in case of failures or dynamic re-optimization in case of performance degradation. It

		will also provide mechanisms for optimization of resources allocation algorithms and schedulers.
UR.14	Proper algorithms for the joint allocation of IT resources in coordination with network resources (L2, transit) SHOULD be available for use by the experimenter	The optimization of the overall energy consumption MIGHT be considered an objective function when taking decisions about the location of cloud resources for data caching and elaboration
UR.15	The overall service offered by the FELIX platform SHOULD be flexible	The flexible term means: (a) dynamic and on-demand usage of network resource; (b) dynamic selection of locations for caching and computing resources depending on experimenters and applications' requirements for QoS metrics, required performances, geographical position, etc.
UR.16	A specific attributes of network connectivity SHOULD be achievable through the intercontinental on-demand network services in FELIX test-bed (for effective data transfer from processing points to caches).	The transatlantic connection between Japan and EU should be capable of transfer at least 10Mbps and target 200ms RTT. It is suggested that capacity will be about 1Gbps circuit, in order to minimize the time of data transfer at long distances.
UR.17	The FELIX platform SHOULD enable network resiliency features	The resources protection and/or restoration will be a requirement for production services based on FELIX framework. In case of a failure (either type of resources – computing, storage, or network) the system should be able to reconfigure the slice in order to assure its availability, with minimum impact to end users. This feature require monitoring features to be running. In specific FELIX: • MAY reroute an NSI connection • MAY move traffic to a backup NSI connection • MAY manipulate SDN flows configuration in order to resolve failures Some use cases MUST have connections re-routing feature in order to be executed properly.
UR.18	The FELIX platform SHOULD provide mechanisms for resource scheduling in future time intervals	FELIX framework should not only allow to immediately (at the moment of request arrival) set up the virtual slices, but also to schedule the configuration in the future. The framework can either postpone the request to be processed at specific time in future, basing on best effort approach, or preferably book the required resources in advance (guarantee their availability at the moment of slice creation). Since advance reservation is a complex problem, especially in dynamic environment, this problem needs to be investigated in details. It is not critical to operation of FELIX environment, yet it will significantly increase its usability.
UR.19	FELIX SHOULD be able to accept flow specific attributes from requesting application (e.g.	FELIX framework should allow end users to define their constraints for resources selection. E.g. for an SDN slice network, a user may want to specify particular flow suggestions, by

	capacity, end points, MAC addresses, IP addresses, protocol type, etc.)	entering network capacity, end points, MAC addresses to be used/connected, IP addressing, protocol types, etc. This values should be an input to FELIX dynamic resources allocation algorithms.
UR.20	FELIX SHOULD provide Accounting features with proper usage policy	FELIX framework despite of Authentication and Authorisation of users should also provide mechanisms for accounting of resources usage, user activity, and also option to respect defined policies regarding resources allocation.

Table 3: SHOULD FELIX users' requirements set

ID	Requirement	Description
UR.21	The FELIX platform MAY support mechanisms for provisioning of point-to-multi-point services, in terms of network connectivity, data delivery and caching	The Use Cases expressed in this deliverable may require an option to send the same data to multiple points. The SDN infrastructure and its controller usually allows such network configuration, however the NSI based long range connections will require additional configuration. NSI protocol currently operates as point-to-point connection service. Allowing multipoint connectivity requires extension of the protocol and enabling additional features at data plane.
UR.22	FELIX framework MAY provide an API for measure and monitoring resources	In order to integrate FELIX with various multiple measurement systems, there is a need to deliver a standardized API. FELIX MAY provide API at least for perfSONAR measurement system, which allow to re-use of existing measurement solution in the created infrastructure.
UR.23	FELIX framework MAY provide mechanisms to synchronize data	Some Use Cases require mechanisms of data synchronization among multiple sites. FELIX can use external tools to deliver such synchronization or incorporate their own algorithms/tools for this, in order to obtain more control. Data replication is however not a critical issue for FELIX to operate.

Table 4: MAY FELIX users' requirements set

4 Summary and Future Work

This document presented initial wide set of FELIX user scenarios alongside the corresponding user requirements for the FELIX virtual infrastructure and overall system. In short, six use cases have been detailed, further grouped into two major groups (Data Domain and Infrastructure Domain) to better reflect what we think is their primary applicability area and stakeholders.

The FELIX Data Domain use cases mostly target the application area of SDN and dynamic interconnections via NSI to improve and innovate data transfers and consumption among geographically dispersed testbeds across continents. Data caching, fast delivery, streaming and the related workflow management are key in this group of use-cases.

On the contrary, the FELIX Infrastructure Domain use cases focus more on the efficient use of federated and dispersed FI resources (in different continents), to migrate entire workloads (VMs and data) or infrastructures in a more efficient way (e.g. with energy saving targets) and enhanced features (e.g. data/service survivability in case of disasters).

From the user’s perspective, all presented use-cases apply to the same and unique FELIX framework architecture, which has to include the common system functionalities derived from specific use cases and users’ goals. The current list of use-cases along with the subsequent user requirements are not meant to be exhaustive. In fact, they represent the initial structured outcome of the activities in WP2 and a preliminary input to the FELIX architecture definition.

Based on D2.1 results, the work of the FELIX partners will proceed to focus more on the architecture design phase. During this activity we will:

- Consolidate the user requirements identified via the use cases, and translate them into corresponding architecture and functional requirements for the FELIX systems;
- Use the full sets of requirements to specify the FELIX architecture and the interfaces and behaviours of the FELIX functional entities;
- Assess the criticality and priority of the different requirements to define an implementation plan that could allow an incremental delivery of functions on the FELIX FIRE infrastructure.

The collection of the aforementioned work phases will be documented in the next project deliverable, i.e., D2.2.

5 References

[1] S. Brander, "Key words for use in RFCs to Indicate Requirement Levels," March 1997. [Online].

[2] Monga, E. Pouyoul i B. Tierney, „Dynamic creation of end-to-end virtual networks for science and cloud computing leveraging OpenFlow/Software Defined Networking," w *Terena Networking Conference*, 2012.

[3] „Cloudstack Zone reference:," [Online]. Available: http://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.1.0/html/Admin_Guide/cloud-infrastructure-concepts.html.

[4] „OpenStack Zone references:," [Online]. Available: http://docs.openstack.org/trunk/openstack-ops/content/scaling.html#segregate_cloud, https://wiki.openstack.org/wiki/MultiClusterZones .

[5] FP7 ICT Project GEYSERS, „"Generalised Architecture For Dynamic Infrastructure Services", Large Scale Integrated Project Co-funded by the European Commission within the Seventh Framework Programme, Grant Agreement no. 248657,," [Online]. Available: http://www.geysers.eu/.

[6] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2001.

[7] Qureshi, Asfandyar et al., „Cutting the Electric Bill for Internet- scale Systems," w *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication - SIGCOMM '09.*, Barcelona, Spain, 2009. 123. Copyright c2009 ACM, 2009.

6 Acronyms

AAA	Authentication, authorization and accounting
API	Application programming interface
BCP	Business continuity planning
CMS	Cloud management systems
CPU	Central processing unit
DC	Data center
DWDM	Dense wavelength division multiplexing
FED4FIRE	Federation for FIRE
FI	Future Internet
FIBRE	Future Internet Testbeds Experimentation between Brazil and Europe (FP7 ICT)
FIRE	Future Internet Research & Experimentation
GEYSERS	Generalized Architecture for Dynamic Infrastructure Services (FP7 ICT)
GLIF	Global lambda integrated facility
HaaS	Hardware as a service
HPC	High-performance computing
IaaS	Infrastructure as a service
I-CSA	Inter-cloud service agent
ID	Identfication
IT	Information Technology
JGN-X	Japan Gigabit Network X (4th generation JGN)
KVM	Kernel VM
L2	Layer two
L3	Layer three
MPLS	Multiprotocol label switching
NREN	National research and education network
NSI	Network service interface
NSI-CS	NSI connection service
NW	Network
OCF	OFELIA control framework
OF	OpenFlow
OFS	OpenFlow switch
OFELIA	Openflow in Europe: Linking Infrastructure and Applications (FP7 ICT 258365)
OMF	cOntrol Management Framework

QoE	Quality of Experience
QoS	Quality of service
RFC	Request for Comments
RISE	Research Infrastructure for large-scale network Experiments
RSPEC	Resource specification
RTD	Round-trip delay
RTT	Round-trip time
SDN	Software defined networking
SFA	Slice-based federation architecture
TCP/IP	Transmission control protocol/Internet protocol
UC	Use case
UPS	Uninterruptible power supply
VM	Virtual machine
VPN	Virtual private network
WAN	Wide-area network

The scientific/academic work is financed from financial resources for science in the years 2013 - 2016 granted for the realization of the international project co-financed by Polish Ministry of Science and Higher Education.