FEDERATED TEST-BEDS FOR LARGE-SCALE INFRASTRUCTURE EXPERIMENTS

FELIX EU-JP

Collaborative project co-funded by the European Commission within the Seventh Framework Programme

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:	39 months

Deliverable D4.3
FELIX Experiments Report Update

Version 1.2

Due date:	31/03/2016
Submission date:	30/04/2016
Deliverable leader:	iMinds
Author list:	Vicent Borja Torres (iMinds), Brecht Vermeulen (iMinds), Carolina Fernandez (i2CAT), Bartosz Belter (PSNC), Łukasz Ogrodowczyk (PSNC), Damian Parniewicz (PSNC), Umar Toseef (EICT), Gino Carrozzo (NXW), Roberto Monno (NXW), Atsuko Takefusa (AIST), Jason Haga (AIST), Tomohiro Kudoh (AIST), Takatoshi Ikeda (KDDI), Jin Tanaka (KDDI).

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 report contains information with the final picture of the FELIX infrastructure deployed and a final revision of the requirements proposed at the initial stage of the project . Also the final results obtained from running the experimental use cases proposed on D4.2 using FELIX infrastructure deployed and final conclusions about the tools, experimental infrastructure. FELIX infrastructure is been developed based on the initial proposed requirements presented on D2.2 as final report a review of the completed requirements is described. This report is an extension of the D4.2 and is the final report that completes work for WP4.

Project:	FELIX (Grant Agr. No. 608638)
Deliverable Number:	D4.2
Date of Issue:	30/04/2016

Executive Summary

The FELIX workpackage WP4 has been responsible for the activities related to the infrastructure maintenance and most important the deployment and execution of the Use Cases.
This deliverable provides a report of FELIX testbed and experimental activities carried out by the partners during the third and final year of the project. This activities have been focused on the Use Cases deployment and execution in order to validate the FELIX Management Stack developed within WP3. During this activities the whole team has been learning and improving as a result of working to reach the goal to successfully execute the described Use Case experiments on D4.2. This is an important goal and asset to explain the approach and commitment of the FELIX project and in extension to FIRE community.

Table of Contents

Figure Summary

Table Summary

1 Introduction

Deliverable 4.3 details the final report of the WP4 experimentation phase of the FELIX project.

The deliverable presents a review of the initial proposed requirements described on D2.2 and is aimed to report which of this requirements are satisfied and which ones not, on the final picture of the FELIX project. Also is presented the final results of the experimental execution of 3 merged use cases described on D4.2 and the conclusion besides the use of the FELIX infrastructure deployed to execute the experiments. The documents is mainly structured in with two chapters; Chapter 2 report on the FELIX Infrastructure Requirements and Chapter 3 reports on the FELIX Use Cases execution and describes for each use case its final deployment, the experiment execution experience and the analysis of the use case results using FELIX infrastructure.

This document is the final report and an extension for D4.2, behind the related work on WP4 and concludes the experimental phase of the FELIX project.

2 FELIX Infrastructure Requirements Report

2.1 System Requirements and Components Implemented

The following tables describe the system requirements defined at the beginning of the project and published on D2.2 on M9. The objective is offer a functional overview of the capabilities of the FELIX framework within the infrastructure deployed.

The tables are structured by different columns:

- Requirement: nominal identifier of the requirement
- Description: explanation of what is exactly required
- Components: developed modules that implement the requirement
- Testing Procedure: which Use Cases triggers the defined requirement
- Comments: additional information

The Use Cases are identified as follows:

- UC1-2: Data Pre-Processing On Demand *(merged Use Cases)*
- UC3: High Quality Media Transmission over Long-Distance Networks
- UC4-5-6: Infrastructure Domain *(merged Use Cases)*

Requirement	**Description**	**Components**	**Testing Procedure**	**Comments**
User request acceptance	The FELIX framework MUST accept user requests for a slice, which includes a minimal set of details required to create a multi-technology multi-domain slice. The "minimal set of details" is technology dependent and is not in scope of this document. The approval of a single request may be the subject of additional considerations according to set policy or system constraints.	M/RO, CRM, SDNRM, SERM, TNRM	UC3, UC4-5-6, UC1-2	

Implement a slice	The FELIX framework MUST implement a user slice, according to resources availability and various constraints, respecting slice requirements provided by a user. The slice may be multi-technology and multi-domain. FELIX MUST be able to orchestrate the resources allocation process and reply user with success or failure notification of slice creation.	M/RO	UC4-5-6, UC1-2	
Request API	The FELIX framework MUST provide a user API, which allows to perform the minimal set of operations: (i) request a slice; (ii) check slice status; (iii) request a slice termination	M/RO, CRM, SDNRM, SERM, TNRM	UC3, UC4-5-6, UC1-2	
Distributed environment support	The FELIX framework by default is a multi-domain environment and therefore its components MUST be designed and implemented in a way allowing the distribution of the components and independent system deployments in particular domains. All framework entities MUST be able to be instantiated as standalone components, and MUST NOT depend on each other. The only allowed dependency is a network communication between entities, however entities should handle communication failures and continue operation without critical errors.	M/RO, CRM, SDNRM, SERM, TNRM, M/CBAS, M/MS	UC3, UC4-5-6, UC1-2	
User slice control	A user MUST have immediate access to a created FELIX slice, so that he/she can reach and manipulate any resource, that was expressed as a slice requirement. FELIX may hide some slice infrastructure components, which are vital for slice delivery but was not explicitly mentioned by a user, e.g. network hardware for inter-domain connectivity, hardware virtualization platform, etc.	M/RO, CRM, SDNRM, SERM, TNRM, OMNI, jFed	UC3, UC4-5-6, UC1-2	
User web portal	The FELIX framework MUST provide end users with a graphical user interface, e.g. a web portal, as an interaction mechanism. A user must be able to manage his slices and reservations through the GUI and receive notification from the system.	jFed	Integration FMS-jFed	

Command line access	The FELIX framework MUST provide users with command line style interface, in order to request and manage slices. It is advised to reuse existing CLI tools, like OMNI or SFI, with proper adaptation to FELIX architecture.	OMNI	UC3, UC4-5-6, UC1-2	
Complex slice infrastructure creation	The FELIX framework MUST be able to build on demand a complex slices, using infrastructures and resources of different domains at the same time. Users should not be restricted by the architecture in the amount of resources allowed to build a slice, except for authorization, policy and infrastructure constraints. Slices may consist of any amount of SDN, transport network and/or IT resources, providing users with wide range of services and giving the control over the resources directly to a particular user. The exemplary usage of FELIX framework is described in [FELIXD21] deliverable in Use Case section.	M/RO		Resources are bound to locations; no scale out allowed by this paradigm
Messaging consistency and integrity	The message exchange between the FELIX framework entities MUST be assured to be secure and consistent, in the sense that message delivery must be controlled and monitored. The entities must be assured that the message is delivered or a failure has occurred. The integrity of the message must be protected, in order to prevent modification of the message content by unauthorized external entities.	M/RO, CRM, SDNRM, SERM, TNRM	UC3, UC4-5-6, UC1-2	Messages are consistent, but not secured against forgery
The FELIX framework must control resources of different types	In order to deliver to a user a multi-technological slice, including resources of different types, the FELIX framework MUST be able to manage different kind of technologies, including SDN, transport networks, and IT resources. FELIX MUST provide mechanisms to request configuration and synchronize technological parts of slice.	M/RO, CRM, SDNRM, SERM, TNRM	UC4-5-6, UC1-2	
Scalable technology modules	The technology management modules (Resource Managers - RM) in single domain MUST be able to be deployed in scalable and efficient way. The particular technological sub-domains managers	CRM, SDNRM, SERM, TNRM	UC3, UC4-5-6, UC1-2	

deployment	should be able to be deployed independently and should not relay on each other during operation, unless a synchronization effort is needed. Therefore an administrator should have an option to deploy only some of the available RMs, and not all of them, if they are not required.			
Support for SDN resources	The FELIX framework MUST be able to configure SDN resource types within a particular domain, in order to create and configure a user slice, and deliver this slice under user control.	SDNRM	UC3, UC4-5-6, UC1-2	
Support for Transport Network resources	The FELIX framework MUST be able to configure a transport network resource types within a particular domain, in order to create and configure a user slice, and deliver this slice under user control.	TNRM	UC4-5-6, UC1-2	
Support for IT resources	The FELIX framework MUST be able to configure an IT (e.g. servers, data storage, etc.) resource types within a particular domain, in order to create and configure a user slice, and deliver this slice under user control.	CRM	UC4-5-6, UC1-2	
Orchestration and synchronization of resources configuration	The FELIX framework MUST orchestrate and synchronize configuration of particular sub-domain configurations (SDN, transport networks, IT, etc.), providing unified slice resources able to collaborate in a transparent way (not disturbing end user actions). Particularly, the integration of SDN and Transport Network resources is critical for achieving the FELIX project objectives.	M/RO	UC4-5-6, UC1-2	
Organized orchestration layer of FELIX framework	The orchestration layer of FELIX MUST be hierarchical, organized, and scalable in the context of deployment and management. This will require coexistence of multiple orchestration entities, able to collaborate in organized manner for: ▪ delegating (also splitting) requests to appropriate orchestrators or RMs	M/RO	UC4-5-6, UC1-2	

	▪ forwarding messages in hierarchical model ▪ synchronize state for consistent global view of slices and resources			
Resources allocation mechanism in distributed environment	The FELIX framework MUST have a resources allocation mechanism, which will be able upon a use request to: ▪ identify required resources ▪ locate the resources in domains (relaying on available information) ▪ construct initial draft of slice description, including information on domains and their resources required to build a user slice ▪ accept constrained queries providing slice descriptions excluding particular resources in a domain (e.g. when a reservation fail due to particular domain, a new resources search should not try to use the ""refused"" resources)	M/RO	UC4-5-6, UC1-2	
Support for inter-domain transit network configuration	Transit Network resources MUST be able to be configured involving multiple domains, so that a created connection can link and/or pass multiple administratively independent domains, managed by different ROs and RMs. The inter-domain configuration must be under control of proper ROs and RMs, which should be aware of such action and required interaction.	M/RO, TNRM	UC4-5-6, UC1-2	
Authentication of users	The FELIX framework MUST be able to authenticate end users in advance, before providing them access to any FELIX controlled resources and slices.	M/RO, CRM, SDNRM, SERM, M/CBAS	UC4-5-6, UC1-2	
Authentication of FELIX entities	All FELIX entities MUST be able to authenticate each other (e.g. with X.509 certificates), in order to prevent unauthorized resource manipulation.	M/RO, CRM, SDNRM, SERM, M/CBAS		Except TNRM and M/MS
Authorization of users	The FELIX framework MUST be able to authorize end users on particular resources usage in order to prevent abuse of resource usage. Users should have access only to the resources assigned to them by FELIX framework. Authorization data will also prevent users from invoking unauthorized actions	M/RO, CRM, SDNRM, SERM, M/CBAS	UC4-5-6, UC1-2	

	and define users roles (e.g. administrator, experimenter, etc.)			
Authorization of FELIX entities	All FELIX entities MUST be able to authorize each other , in order to prevent unauthorized resource manipulation.	M/RO, CRM, SDNRM, SERM, M/CBAS		Except TNRM and M/MS
Users notifications	Users MUST be notified about significant events happening in the FELIX framework environment. Such events include: ▪ acceptance of a single slice request ▪ confirmation of resource booking ▪ notification on slice set up ▪ notification on recognized failures ▪ confirmation of slice tear down ▪ notification of administrative messages (e.g. planned maintenance) The notification may be delivered at least as: ▪ email sent to a user, if a user explicitly express such interest while request submission and provide system with a valid email address ▪ a notice in GUI related to a user, a slice or a particular reservation.	M/RO, CRM, SDNRM, SERM, M/CBAS, OMNI, jFed		All events but administrative are provided by manifest of M/ROs and RMs
Resources awareness	The FELIX framework MUST be aware of all resources available for configuration in the whole controlled environment. This information is vital for proper operation, resource management and configuration. This information must be organized and structured allowing browsing resources types, domains in charge of them, and relation between domains (e.g. network connectivity).	M/RO	UC4-5-6, UC1-2	Informationfor resources, domains and peers in other domains are kept up-to-date in M/RO DBs
Resource information propagation	The global resource information MUST be dynamically distributed to all FELIX entities, which consider this information as critical to operate (mostly ROs and resources allocation entities). The framework MUST deliver an automated mechanism (e.g. a lookup service) where resource information can be stored and distributed in consistent state.	M/RO, CRM, SDNRM, SERM,TNRM		Resource status distributed from RMs to ROs, from ROs to MROs; periodicallyand upon resource

				request
Resources usage tracking	The FELIX framework MUST keep track of resources usage, assignment, and availability in order to search and allocate only free resources to new slices. FELIX MUST guarantee that the same resource is not shared between more than one user at the same time, providing exclusive resources access and isolation of slices.	M/RO	UC4-5-6, UC1-2	
End users VPN service	The FELIX framework MUST provide a VPN service with configurable resource access, limited to slice scope. This will be one of the mechanisms providing isolation of user slice and default way for users to access their slice resources. The configuration of VPN service per user must include setting up authentication and authorization details, restricting access to allowed resources only, setting up access policies, and possibly firewall restrictions. The user must be authenticated and authorized to the VPN service in order to prevent resources abuse or unauthorized access. The VPN service is configured on per slice and per user basis, and the configuration process is allowed to be manual.			Access to slices' resources controlled by filters (e.g. VLAN) in network RMs and by PKI in CRM

Table 1: MUST System Requirements

Requirement	Description	Components	Testing Procedure	Comments
Accounting information	The FELIX framework SHOULD be able to collect accounting information in order to track user activity and resources utilization. This will allow create resources usage reports and account users for their resources utilization. Accounting information may also be used to restrict users e.g. with resources quotas, etc.	M/RO, M/CBAS	UC4-5-6, UC1-2	
Monitoring up-to date resources status	The resources information and/or FELIX system entities SHOULD be able to be updated frequently or on-event by monitoring system, in case of resource status change, i.e. in unexpected failure conditions. This will potentially allow to make FELIX framework	M/RO, M/MS	UC4-5-6, UC1-2	MMS updated with slice info upon each provision & delete

	more robust and react on critical situations.			
Notify users on unexpected failures	The FELIX framework SHOULD be able to notify users about critical situation encountered during slice operation, i.e. unexpected resources failure. The system may or may not undertake a repair action, however user SHOULD be notified that the slice is not fully operated or unavailable at all. The notification SHOULD include the amount of information will not discover FELIX critical information, yet it will be meaningful for the end users, allowing to understand the cause and location of the problem.	M/RO, CRM, SDNRM, SERM,TNRM, OMNI, jFed		Users are notified after resources failure, but not about lack of fully operational slice
Resilient service configuration	While searching, allocating and configuring FELIX resources for slice purposes, FELIX framework SHOULD be able to implement resiliency features, which will protect all or critical resources in case of failure. Resiliency SHOULD be optional for end users and must be explicitly expressed by an end user at slice request time.			
Critical failure restoration	In case of a critical failure regarding usage of slice resources, the FELIX framework SHOULD take a repair action. If a user expressed resiliency expectation, the protected/backup resources can be used. If a user did not express resiliency as a requirement, or failure applied to non protected resources, a repair action MAY involve reconfiguration of existing slice pre-empted by now resources search with additional constraints.			Improved error handling on resource management was added
Optimization and automation of resources allocation	The resources information and/or FELIX system entities SHOULD be able to be updated frequently or on-event by monitoring system, in case of resource status change, i.e. in unexpected failure conditions. This will potentially allow to make FELIX framework more robust and resilient, and as a consequence provide better, more reliable services to the end users. The information on current and planned resources status can be potentially also used for resource scheduling and planning, in order to make the	M/RO, M/MS	UC4-5-6, UC1-2	

	assignment more stable and failure proof.			
Dynamic resources configuration	The FELIX framework resources allocation mechanism SHOULD be automated, so that the incoming request are served immediately without delays of human intervention. Framework should have autonomy to decide which resources and in which domain should be delegated in order to build a user slice. The process should include optimization mechanisms and respect user constraints and SLA.	M/RO, CRM, SDNRM, SERM,TNRM	UC4-5-6, UC1-2	Delegation occurs only to proxy resources to their proper management area
Automated configuration of VPN service	The FELIX framework VPN service SHOULD be configured automatically by FELIX framework on per slice per user basis, regarding a user slice request and assigned resources. The framework should have all credentials to provide authentication and authorization configurations, set up firewall policies, user policies, access privileges, and whatever other actions, required to allow particular user to access his assigned resources/slice. At the same time the configuration must assure slice and users isolation in the FELIX environment, preventing resources abuse and over-utilization.			
RMs may be able to interact directly in inter-domain environment	For optimization and efficiency of configuration process RMs SHOULD be permitted to interact, despite they may be deployed in different administrative domains and be supervised by different ROs. In case a configuration of one resources depends on configuration of other resources in different domain, and those resources are of the same type, the RMs can interact directly, without intermediate supervising ROs. This situation can occur e.g. while setting up transport network connectivity and domains must agree common VLAN identified or exchange other connection specific attributes.			Horizontal communication allowed in M/RO, M/MS and M/CBAS (not RMs)
Resources allocation optimization	While searching and allocating resources, the FELIX framework SHOULD use mechanisms allowing optimization of resources utilization, which are not explicitly mentioned in a user request. Such mechanism may e.g. consider load balancing,			

	resources utilization levels, overall energy consumption, etc.			
Request attributes	A user slice request must contain a minimal required set of attributes as defined by FELIX. The FELIX framework SHOULD however allow to specify also optional attributes, which constraint a slice resources in more advance manner. For transport network connection a minimal set of information to deliver a circuit is a start point, end point and capacity, while user can additionally request e.g. RTT limits, or allowed VLAN range. For SDN resource types, a user may want to define details regarding traffic organization specifying particular network flows or restrictions on SDN resources.	M/RO, CRM, SDNRM, TNRM	UC4-5-6, UC1-2	User can provide optional attributes in M/RO (virtual links), CRM (template, server), SNDRM (filters), TNRM (capacity, vlan tag and range)
Default slice controller	The FELIX framework SHOULD deliver a default slice controller, which can be used by a user to manipulate resources within a particular slice. A user however is not obliged to use this controller and may deploy its own. The decision must be however taken at the slice request submission time, as proper resources holder will be prepared.			
Advance reservations	The FELIX framework SHOULD support advance reservation scheduling, where users can specify a slice start time in the future and a lifetime duration of a slice. This will force the framework to analyse resources availability not only in the moment of serving the request but also future planning of resources usage. The time constraints for user (e.g. the duration of a reservation and how far in advance can a reservation be requested) should be defined in the form of service policy and framework			Not compatible with current paradigm

Table 2: SHOULD System Requirements

Requirement	Description	Components	Testing Procedure	Comments
Multi-point to multi-point network connectivity	The FELIX framework MAY support multi-point network connectivity for slice building. The default method for implementing a transport network connectivity, i.e. for multi-domain purposes, is a point-to-point service. Providing multi-point services will enable more scalable configuration and allow more advanced slice configuration.			Multi-point was considered in NSI but left to improve intelligent mapper on end-to-end link creation
Data replication mechanism	Usage of multiple data storage facilities in single slice may potentially require a synchronization of data repositories. The FELIX framework MAY support such synchronization and provide data replication tools internally.			
Monitoring API	The FELIX framework MAY implement API of various monitoring tools (e.g. PerfSONAR, Nagios, etc.), which will allow to reuse existing monitoring solutions and improve FELIX monitoring capabilities.	M/MS	UC4-5-6	

Table 3: MAY System Requirements

After this analysis of the requirements implemented and deployed throughout the project we can observe that most of the hard (*MUST*) requirements have been provided and tested first manually, then assessed through the merged Use Cases. Some of the soft requirements (*SHOULD* and *MAY*) have been implemented as well. Those requirements lacking may imply either a considerable deal of technical complexity or were considered not aligned with the updated vision of the projects and the objectives we should pursue; i.e. a framework able to abstract complex inter-domain connections through different types of transit networks.

2.2 International Connectivity

The FELIX test-bed leverages international connectivity through the TN-RM. The TN RM sends a path request to an NSI Aggregator, which subsequently sends the request to the related provider agents (PA) on the GLIF AutoGOLE test-bed establishing an inter-SDN island path. The GLIF AutoGOLE test-bed has over 20 participating R&E network domains from around the world, such as Netherlight, GEANT, iCAIR (StarLight), and JGN-X. These organizations help maintain AutoGOLE on a volunteer basis and it is expected that the AutoGOLE test-bed will support Large Hadron Collider (LCH) experiments in the near future.

FELIX is 1 of 6 early users of the AutoGOLE test-bed. Because of this, we encountered some stability issues, however, we were able to inform the AutoGOLE community how to improve the test-bed. The large network of people in AutoGOLE, is working very hard to improve the stability of the test-bed and our contributions as a user were invaluable to their efforts.

Although an NSI-based path over the AutoGOLE test-bed is of higher bandwidth and quality than the Internet, the global distribution of the network domains can create a significant time-lag when requesting assistance from the different network operations centres (NOC) primarily due to differences in time zones. It is not feasible to have NOCs operate 24 hours a day, 7 days a week, so regional time differences should be considered when establishing international connectivity. In spite of this, all NOCs we contacted resolved issues in a timely fashion and were easy to work with.

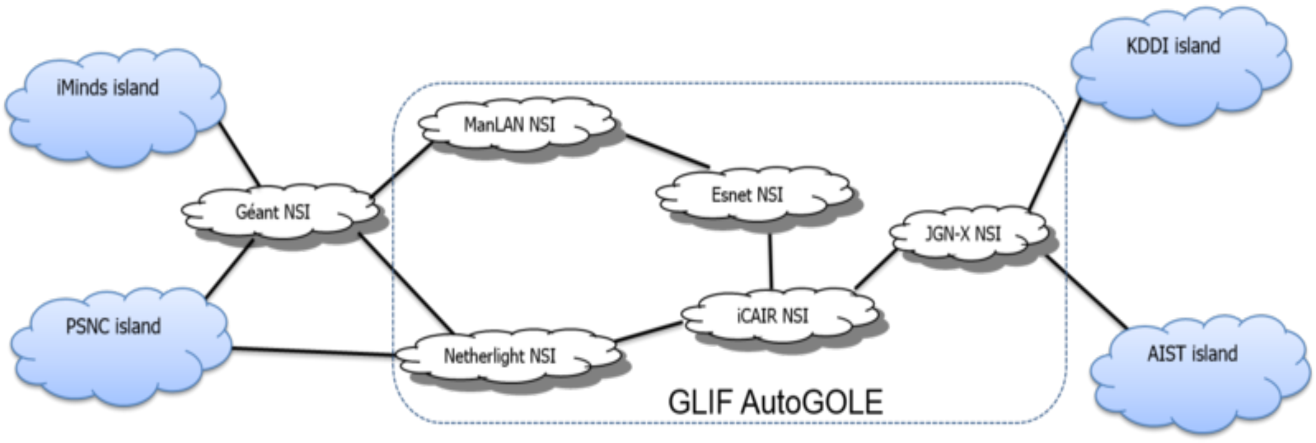


Figure 1: International Data Plane Connectivity

3 FELIX Experimental Report

3.1 UC. High Quality Media Transmission over long-distance networks

3.1.1 Description Final Deployment

As already described in the deliverable D4.2 [FELIXD4.2], the goal of the experiment was to examine long distance network capabilities for media streaming, in particular to evaluate user experience (QoE) with high quality media encoded using H.264 with different bitrate and to examine a new intelligent network application for controlling high quality media streaming.

The deliverable D4.2 [FELIXD4.2] presents also initial plans of software ecosystem, infrastructure requirements and execution workflow of the High Quality Media experiment. The experiment environment was deployed within FELIX infrastructure (in PSNC and AIST) and it was composed of software which was implemented by PSNC and AIST.

- GUI used as front-end for demonstration of experiment
- Proxy module implementing back-end mechanisms required by the experiment and other existing software which had be only properly configured.
- UtraGrid-Media-Streamer providing media content streaming.
- UltraGrid-Media-Player capable of displaying 4 media streams at once
- Ryu as SDN-Controller controlling OpenFlow switches within FELIX infrastructure
- PSPacer responsible for rate limitation control over a link

GUI, Proxy module and PSPacer were implemented from scratch. The GUI and Proxy modules for purpose of High Quality Media Transmission use-case was developed using FELIX GitHub [FELIXGHUB].

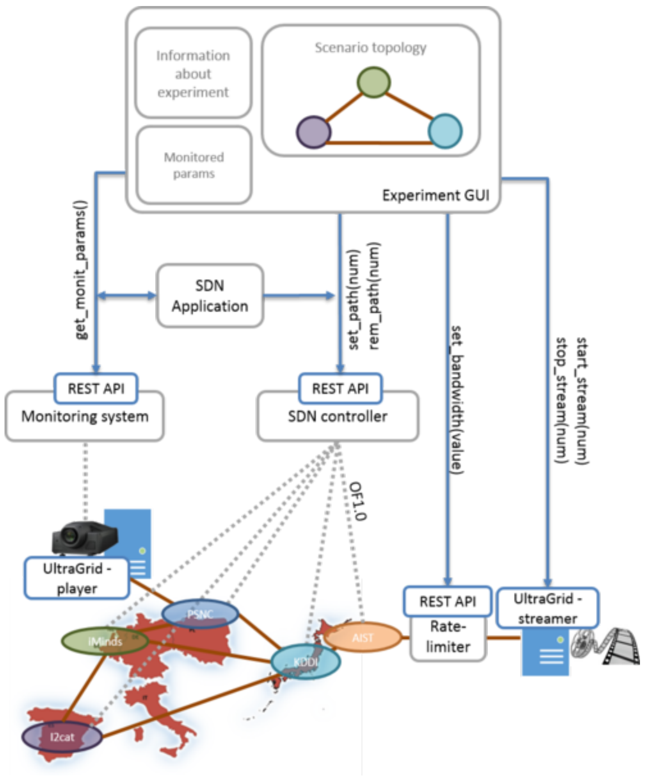


Figure 2: High Quality Media Transmission over long-distance networks experiment overview

The experiment was originally planned to utilize all FELIX infrastructure resources, use NSI connection, static links, etc. The implementation of that experiment, which was FELIX use-case implemented much earlier than rest of FELIX use-cases, was a trigger to all FELIX island operators to ensure that all equipment and patchcords are in place and FELIX stack components are deployed and configured. However, during the experiment preparation, we discovered that AutoGOLE NSI connections were not stable enough for us in that time and static links couldn't be used because of discovered misconceptions in communication between RO and SERM modules. The above mentioned issues were starting point for the FELIX project to develop concept of GRE-TNRM module, offering L2 GRE tunnels between FELIX island. Although GRE tunnels cannot provide the same quality as an NSI connection (potentially), their usage was crucial for presenting demonstration of that use-case during TNC 2015 conference, where part of PSNC island were deployed on the exhibition booth and GRE tunnel over Internet was the only possible way to interconnect local infrastructure to PSNS and then AIST island.

3.1.2 Experiment Execution

The High Quality Media Transmission over long-distance networks experiment was developed and executed between February-June 2015. The experiment was successfully performed during FELIX EC review meeting and TNC 2015 conference. This experiment was performed as first by the FELIX project and discovered a lot of issues with AutoGOLE infrastructure which weren't stable and important problems in the FELIX Software Stack. Moreover, the high demands of the use case concerned the high capacity network for media streaming. The NSI implementation in the AutoGOLE infrastructure we used was not able to guarantee the requested capacity on the end to end path, due to problems in the data-plane of some domains. Paths reserved in the best effort manner between some NSI domains didn't guarantee stable bandwidth on the links. Performance tests of the links between Poland and Japan showed that there were bottlenecks in the NSI network. It was mainly because traffic conditioning mechanisms are not fully supported by the NSI.

Issues or limitations:

- It was not possible to create any EU-JP connection on time using NSI AutoGOLE infrastructure because of initial problems with:
  - a lot of peering issues (e.g.: between PSNC and Netherlight and between GEANT an MANLAN) and missing or disappearing STPs in various network domains topology databases related to usage of old-style STP entries.
  - certificates to became invalid.
  - computation errors generated by AutoGOLE pathfinder algorithms.
  - rate-limitation not applied by network domain switches to traffic within NSI created connections.
  - lack of QoS and traffic conditioning mechanisms between NSI domains.
- Problems related to slice creations:
  - many misconfiguration of SDNRM and SERM components within facilities.
  - wrong handling of static links by SERM.
  - RO integrations with other FMS components not finished.
  - not clear guides which User Interface and how to use by the experimenter.
- Issues related to UltraGrid software
  - some issues related to FPS parameter in the application was reported to UltraGrid development team

For this experiment usage of NSI AutoGOLE infrastructure was not successful and in March 2015, it was decided that the experiment will be executed with usage of GRE tunnels instead of NSI connections. Any other network and FMS problems were resolved by manual interventions or simplification of the network setup.

3.1.3 Analysis of the Result

In this section we recall the table with the KPIs presented in the D4.2 and we introduce a new column that describes its status (i.e achieved or not). In the case a proper justification is added.

KPI	FELIX approach	Status
Streaming is successfully provided by the media provider and received by the media consumer	The FELIX project provides test-bed facilities and FELIX Control Framework for creation and management of the experimental slice and to run experiments over a long distance network. Additional hardware and software facility such as media streamers, players, rate-limiters can be added to the slice for media streaming through SDN-controlled network.	Done. UltraGrid software was used for streaming and displaying.
Network and streaming monitoring data are collected and affect decisions taken by the network application	The smart network application runs on top of the Monitoring System and SDN Controller and can take a decision about network reconfiguration for transmitted media characteristics improvements.	Done. Streaming traffic rate on network interfaces, round trip delay time between media player at PSNC and media streamer at AIST site, number of frames displayed and number of frames lost were collected within the infrastructure and used by network application.
The network application for the streaming control can handle up to four parallel media streams	The network application for the streaming control can handle up to four parallel media streams	Done.
Interruption of media streaming due to network reconfiguration takes less than 10 seconds	Network rearrangement in the OpenFlow-controlled network is a matter of deleting old and adding new flow_modes to all OpenFlow switches on the path. During the FELIX experiment it is also time needed for rate-limiter module reconfiguration.	Done. The time period is assumed as time of media quality observation. The network reconfiguration (skip to the next path and guarantee the higher bandwidth) was done immediately after decision of reconfiguration.
The provisioned slice provides up to 80Mbps capacity for streaming four parallel media contents each encoded with H.264 @20Mbps	1Gbps is limitation of the NSI-controlled network. However, this bandwidth is not guaranteed in the network and may be shared by other NSI interconnections. Network performance provisioned through NSI should be examined also during the FELIX experiments.	Partially done. NSI connnections wasn't used but GRE tunnels over Internet were capable of transmitting streaming at 80Mbps rate. To limit the bandwidth to proper value (20, 40, 60, 80 Mbps) we used PSPacer software module.

Table 4: KPI UC High Quality Media Transmission Over Long-Distant Networks

3.1.4 Conclusions

Implementation and HQ media transmissions use case showed high demanding on the network infrastructure. In particular we examined how capacity of the network influence the quality of the video encoded with different parameters. Moreover use case demonstrated importance of the traffic conditioning mechanisms implemented in the network for the HQ streaming and that lack of QoS guarantees makes the experiment creation impossible. Preparation of the live transmission from Japan to Poland was possible with support of Media Department of PSNC, especially in the area of media equipment and software installation and configuration - these additional stuff was connected to the FELIX infrastructure just for the demonstration purposes. For the experiment, the distributed FELIX infrastructure was essential to analyse influence of long distance connections to quality of media streaming and hardly possible without FELIX project.

As this experiment was the first to be performed by the FELIX project we discovered a lot of issues with AutoGOLE infrastructure and important problems in the FELIX Software Stack. Looking over AutoGOLE infrastructure problems timeline, the FELIX project required 4 months from the point where the FELIX facilities were able to start requesting AutoGOLE connections to time where those requests were really successful and traffic was exchanged between PSNC and AIST. To enable EU-JP data plane traffic, actions from many network domains in EU, US and JP were required. The FELIX project, as one of the first actual users of the AutoGOLE NSI infrastructure, has revealed many points of improvement, both technological and procedural. The feedback provided during execution of the experiment were crucial for overall FELIX development in terms of both infrastructure and software. Important benefit was introduction of GRE tunnels for FELIX inter-domain networking solutions which was then used by other FELIX use-cases.

Results of that experiments were the following:

- Estimation of the network capacity needed for the transmission of the media
- Development of real time monitoring system of the media traffic to observe possible stream degradations due to traffic congestions on the path
- Implementation of smart network application for automatic network reconfiguration and adjustments during the degradation of parameters
- Development of the *RateLimiter* software module which allowed to emulate the network capacity over the links.

3.2 UC. Infrastructure Domain

As previously described in Deliverable D4.2 [FELIXD4.2], Section 2.2.1.3, the goal of this experiment was part of a combined effort for the Infrastructure Domain use-cases. Three different infrastructure domain use cases were merged into one that would demonstrate and assess the capability of the FELIX system. These use cases share common success conditions and goals with respect to the FELIX project and leverage on the same infrastructure resources and FELIX management stack.

The Infrastructure as a Service (IaaS) migration use case is based on the notion of a Business Continuity Plan (BCP). After the Great East Japan Earthquake in 2011, many organizations are investigating Cloud technologies that facilitate the migration of services to a remote data centre, especially during a natural disaster. These disasters that disrupt the electric power supply require the cooperation between two data centres in order to maintain services. IaaS provides isolated resources in the form of isolated tenants consisting of virtual machines (VM), storage volumes, and networks. Multiple users then create and deploy different business related services on these tenants. In the case of a natural disaster, the entire IaaS including all tenant associated resources, need to be migrated to another remote data centre. The IaaS use case demonstrates this capability of FELIX.

The Data Mobility Service use case provides satisfactory quality of service by an inter-cloud system. In the IaaS migration use case the trigger of migration is the disaster. For this use case, the trigger for migration is based on the user's location. If the user of the service moves to a remote location, such as during a business trip to a foreign country, the user's experience of the service is degraded due to the longer latency between the user and server. To improve this situation, the server is migrated to the nearby cloud automatically by the system for the Data Mobility Service. We employed the virtual desktop service in this use case.

3.2.1 Description Final Deployment

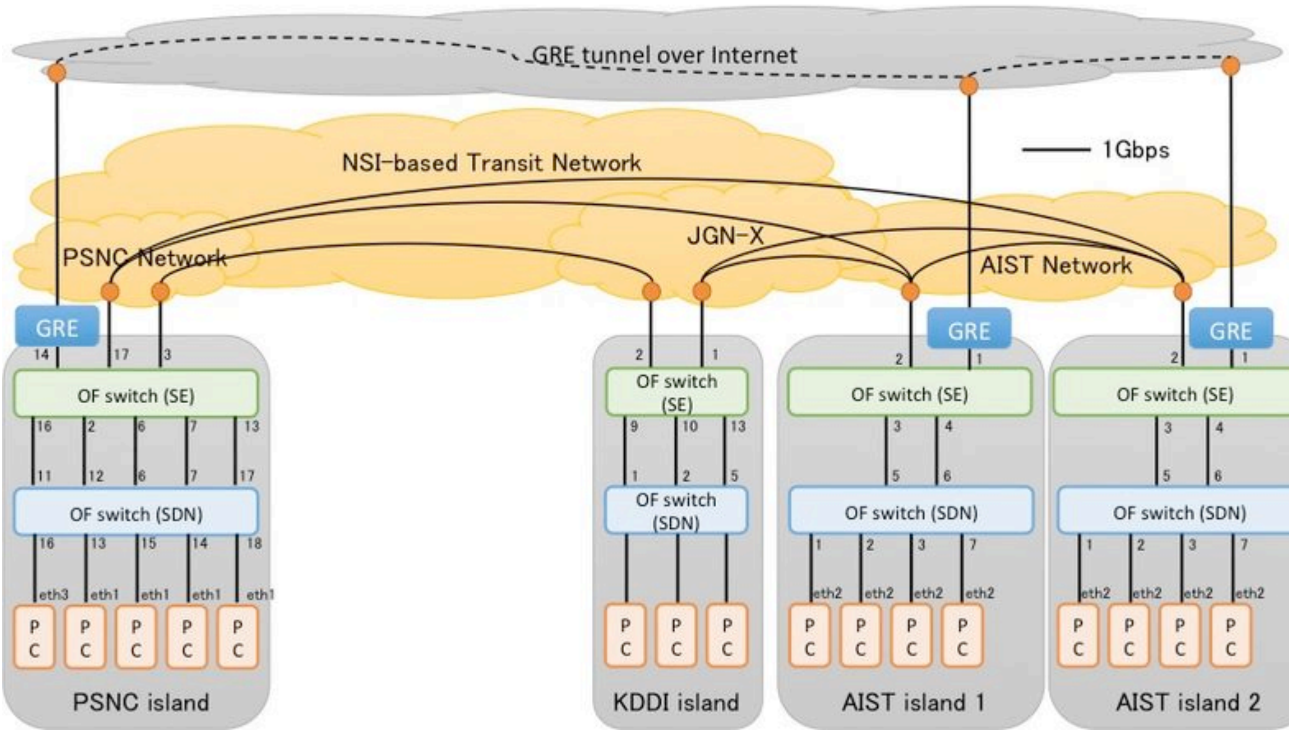


Figure 3: Slice composition of the Infrastructure domain merge use-case

Fig. 2 shows the resources required for the infrastructure domain merged use cases. The slices are allocated over 2 of 4 islands, AIST1, AIST2 and KDDI in Japan, and PSNC in Europe. This set-up allows us to incorporate all the required building blocks from the FELIX Management Stack, i.e M/RO (at both European and Japanese level, as well as at the island level), CRM, SDNRM, SERM and TNRM (NSI and GRE). The required resources such as VMs, openflow switches, stitching entity devices, GRE tunnels or NSI links, and software components are described in Appendixes.

3.2.2 Experiment Execution

For the disaster recovery use case, we performed an IaaS migration over the FELIX test-bed based Hardware as a Service (HaaS). Unlike general HaaS, our proposed HaaS provides users with virtual resources as if they are physical resources. Therefore, the IaaS environment can easily and seamlessly extend onto the HaaS in a different data centre when the resources managed by the IaaS are saturated. This is achieved by the user constructing an IaaS environment over the virtual environment using nested virtualization technologies. The HaaS layer enables the migration of the entire IaaS environment, including user VMs, shared storage, and management servers easily.

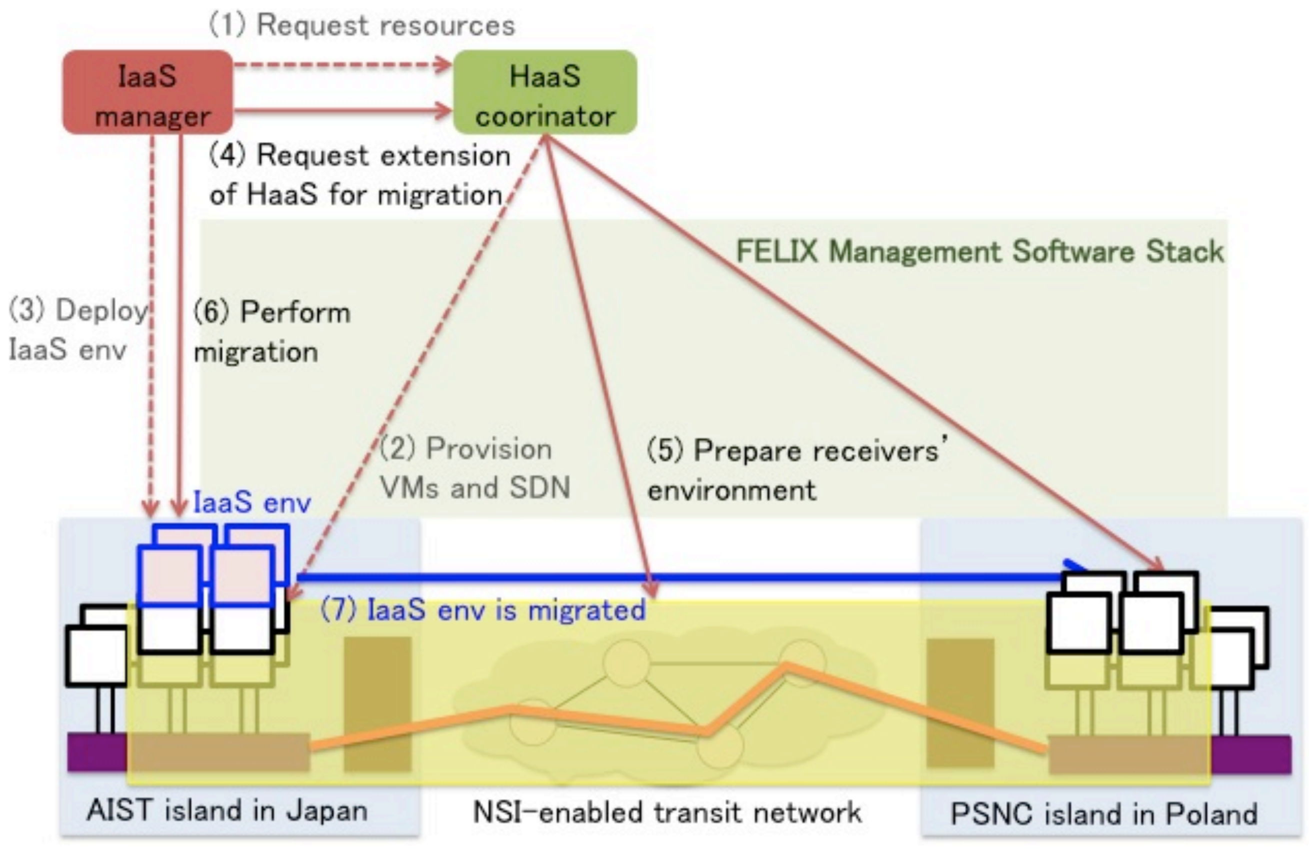


Figure 4: Slice composition of the Disaster Recovery by Migrating IaaS to a Remote Data Centre use-case

An overview of the IaaS migration appears in Figure 3. The migration is from source to remote SDN island, through FELIX using NSI-enabled transit network for slice extension. A VM provisioned by the C RM is called Layer 1 VM (L1VM), and a VM provisioned by a FELIX user, namely IaaS manager, is called Layer 2 VM (L2VM). The steps of IaaS migration consist of:

1. Construction of IaaS in the source SDN island. Based on the IaaS manager's request (1), the HaaS coordinator provisions a slice including SDN links and VMs for IaaS (2). Then the IaaS manager deploys an IaaS environment on the L1VMs in the source SDN island (3). The VMs for IaaS, which consists of one management server VM with storage and user VMs, are deployed as L2VMs.

2. Slice extension to the remote SDN island. When the IaaS manager requests IaaS migration because of a natural disaster (4), the HaaS coordinator prepares VMs and SDN links in the remote SDN island, TN links between the islands, and then extends the IaaS L2 data plane (5).

3. Stop and transfer the IaaS environment. The IaaS manager instructs the management server to hibernate the user VMs and enter IaaS maintenance mode. Then, the disk image of the management server VM is migrated to a L1VM in the remote island (6). 4. Completion of IaaS migration. The IaaS manager restarts the management server and instructs it to resume user VMs on the L1VMs in the remote SDN island.

As for the Data Mobility Service by SDN Technologies Use Case, the Virtual Desktop Infrastructure (VDI) system was deployed over the FELIX test-bed and the experiment was performed. We used a similar infrastructure environment as described for the IaaS migration use case. The experimental slice is created with Computer Resource (CR) in KDDI and PSNC, with NSI-enabled network as the transit network. The user's virtual desktop is running on the CR at the home site. When the user accesses their desktop with VPN from a remote site, the VDI system migrates the virtual desktop to the appropriate cloud based on the user location, using similar steps as outlined above for the IaaS migration. After the virtual desktop migration is completed, the user experience is at the same level of performance as at the home site. Importantly, in contrast to the IaaS migration, the trigger for migration is the remote access of the desktop, with the general experimental execution and infrastructure being the same between the two use case scenarios.

3.2.3 Analysis of the Results

As a means to assess the experimental results, we recall the table with the KPIs presented in the D4.2 and add a new column that describes its status (i.e. achieved or not), with proper explanation as necessary.

KPI	FELIX approach	Status
Resource composition	Slice creation and provision should be directly managed by the FELIX management framework	Done.
Slice creation / modification / destruction time	The FELIX framework allows to automate and reduce time to set up and tear down resources over the different islands	Done.
Interconnectivity between islands using NSI	FELIX framework will be able to define links over combined SDN and NSI networks.	Done. Requests are sent to the related PAs on the GLIF AutoGOLE test-bed for establishing an inter-SDN island path. The AutoGOLE test-bed is maintained on a volunteer basis and consists of

		over 20 participating R&E network domains from around the world, such as Netherlight, GE´ANT, iCAIR (StarLight), and JGN-X.
The FMS maintains the global view of the topology the FELIX infrastructure	FELIX framework will be able to discover/receive the complete topology in order to set up the testing environment for an experimenter.	Done.
Improve the migration efficiency and the QoS the user experiences after migration of services	FELIX framework will provide the resources that are at the optimum position within the global infrastructure.	Done. When the user is in Japan, the migration of user's virtual desktop from EU to Japan was successfully and the user experiences was improved.
Determine additional data centre to support the migration in a specific amount of time if resources are available	FELIX framework will be able to automatically provide compute and network resources that are globally dispersed.	Done.
Perform data migration between two sites within a specific amount of time depending on size of data and infrastructure requirements	FELIX framework will incorporate NSI transit networks with SDN islands to support stable, rapid migration.	Done. The IaaS migration was successfully done in 10 minutes for a 13.7GB image.
User request redirection transparent to the user (i.e., user does not notice changes about serving data centre)	FELIX will redirect traffic between the old and new serving data centres without requiring any configuration changes at the user end.	Done, the user traffic was redirected to migrated user desktop in new location.

Table 5: KPI UC Infrastructure Domain

3.2.4 Conclusions

Both the IaaS migration use case and the Data Mobility Service use case, demonstrate the overall functionality of the FELIX infrastructure and software stack. Both use similar infrastructure setups, but vary with the trigger for the migration of the data centre.

The TN RM we developed successfully managed inter-connections between distributed SDN islands for the IaaS migration and Data Mobility Service use case. The TN RM dynamically provisioned NSI-based paths or GRE tunnels, which provides increased flexibility when establishing connections between islands. We also successfully performed the migration of an entire IaaS environment across intercontinental SDN islands through the FELIX infrastructure, using a nested virtualization approach. Although the overhead associated with KVM nested virtualization technology is not negligible and is not suitable for practical IaaS operations, our results are important performance measurements during an IaaS migration. In the future, GENIv3 will support operations for VM migration and that both nested virtualization technologies and migration technologies will improve, especially using bare-metal Cloud. Importantly, the disk image size of a large-scale IaaS environment will varying depending on the applications and services they are providing. This will directly impact the migration time. These performance considerations were also discovered in the Data Mobility Service use case.

As mentioned in D4.3 International Connectivity section, the FELIX test-bed leverages international connectivity through the GLIF AutoGOLE test-bed for the establishment of an inter-SDN island path. As FELIX was 1 of 6 early users of the AutoGOLE test-bed, there were stability issues, however, we provided valuable information to AutoGOLE community and they are working to improve the test-bed for its users.

3.3 UC. Data Pre-processing on Demand

In deliverable D4.2 [FELIXD4.2] we summarized the basic ideas and goals of two Use Cases originally planned within the Data Domain, namely "Data on Demand" and "Data pre-processing", and compared the similarities between them to explain the motivation behind the decision to merge them into a single one, i.e. "Data pre-processing on demand", which would combine their goals and workflow to fit both scenarios and assess a combined set of requirements. The document also presented the infrastructure resources to be requested to set up the environment, an overview of the architecture of the software to be developed and the procedures to deploy the applications for the Use Case using the FELIX Management Stack (FMS).

The main goals of the merged Use Case can be summarized as follows:

- Demonstrate the possibility of using FMS to allocate and configure the slice over a large-scale environment composed of two islands in Europe and one in Japan.
- Verify the integration of an SDN-based controller to reconfigure network paths dynamically.
- Highlight the combined management of different kind of resources, i.e. computing, OpenFlow, stitching and transport network resources.
- Showcase the transfer of a considerable amount of data through traversing the SDN and TN domains requested for the slice.

During the development and initial deployment stages of this Use Case we introduced some deviations to the original idea; yet assuring it does not affect the objectives or system requirements to be covered by the experiment. One example is the set up of the dynamic connection at the SDN intra-domain level alone, which means the user interacts directly with the SDN controller to insert or delete rules in the switches; instead of setting up the whole intra- and inter-domain connectivity. Though this is perfectly feasible through FMS, we limited to set up dynamic SDN connections for convenience during the development of the custom modules.

3.3.1 Description Final Deployment

Fig. 5 shows the physical resources requested for the slice created for the Data Pre-processing on Demand experiment. The figure does not show the server that hosts the Ryu (SDN) controller, where this server is physically located in PSNC and reaches devices through a VPN.

The slice is composed of 3 islands: i2CAT and PSNC in Europe and AIST in Japan. This set-up allows us to incorporate all the required infrastructures and building blocks from FMS, namely M/RO (at both European and Japanese level, as well as at the island level), CRM (XEN and KVM), SDNRM, SERM and TNRM (NSI and GRE). CBAS is also used for the initial generation of the slice credentials.

The physical resources requested per island are part of the initial environment and are described in the following list:

- i2CAT island: 2 virtual machines (*CRM*), 2 OpenFlow switches (*SDNRM*), 1 stitching entity device (*SERM*) and 1 GRE host (*GRE-TNRM*), which connects each through a tunnel to PSNC and AIST.
- PSNC: 2 virtual machines (*CRM*), 1 OpenFlow switches (*SDNRM*), 1 stitching entity device (*SERM*), 1 NSI endpoint (*NSI-TNRM*) to connect to AIST and 1 GRE host (*GRE-TNRM*) which creates a GRE tunnel to connect to i2CAT.
- AIST island: 1 virtual machines (*CRM*), 1 OpenFlow switches (*SDNRM*), 1 stitching entity device (*SERM*), 1 NSI endpoint (*NSI-TNRM*) to connect to PSNC and 1 GRE host (*GRE-TNRM*) to connect to i2CAT.

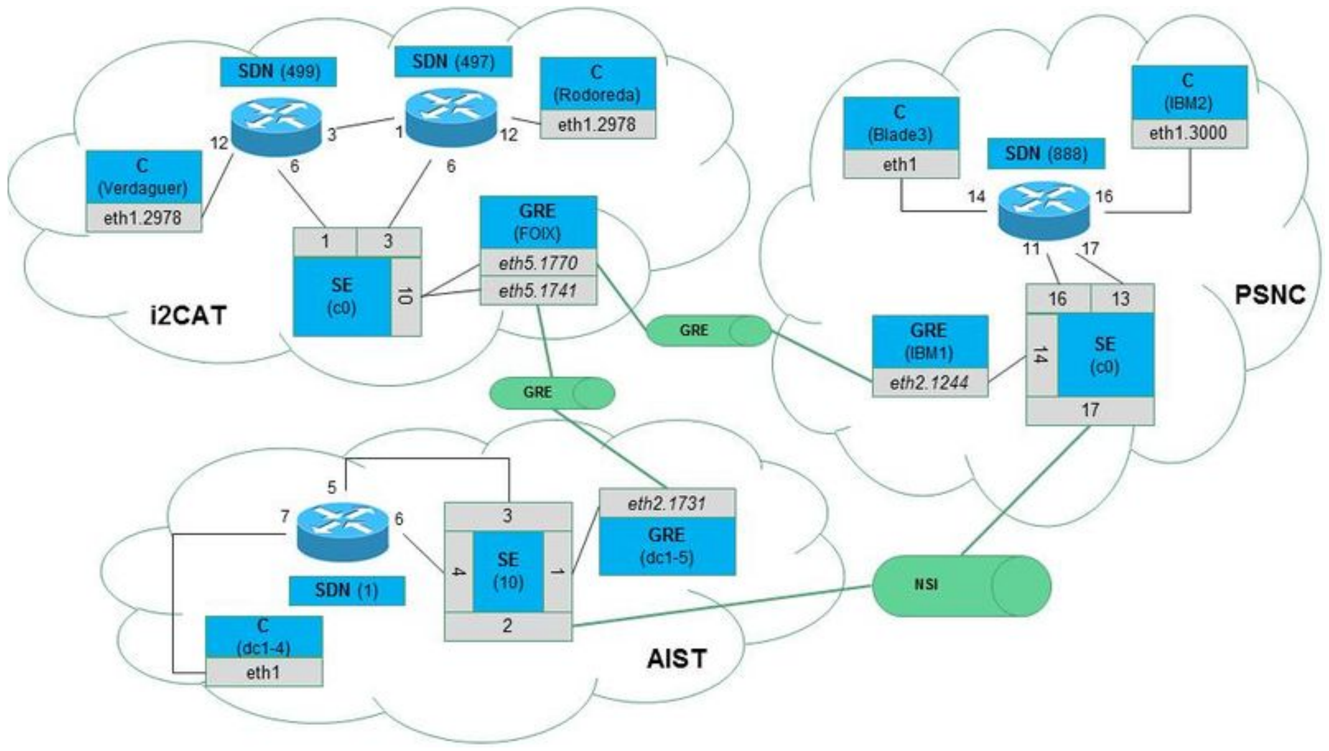


Figure 5: Slice composition of the Data Pre-processing on Demand use-case

Note that for every island the SERM installs flow entries into the physical device in order to enable traffic transmission between local SDN domains and NSI or GRE transit domains that interconnect each island with others in the slice.

Besides the provisioning of the physical resources, the implementation of the Use Case has followed the objectives defined in the introduction by developing a set of custom software tools for the management of the experimental scenario; but also by using the feedback obtained from the deployment of the experiment to extend the FMS with easier ways to reserve resources. The improvements on FMS has been described in the D3.5 consolidated report; and the set of custom tools implemented during the experiment preparation is briefly described here. In short, we used a modular system composed of five basic applications, i.e:

- **Agents**: software dedicated to correctly manage the SMOS (Soil Moisture and Ocean Salinity) data files stored in the Data Centres (DCs). It can compress this data and copy the files into a different location in the slice. Moreover, it can be used to measure the RTT value to any other agent in the slice and report the statistics to the *Collector*.
  - The *Agents* are installed inside specific VMs at i2CAT (servers *Rodoreda* and *Verdaguer*), PSNC (server *Blade3*) and AIST (server *dc1-4*).
- **Collector**: core component of the Use Case that reacts to the REST requests. It performs basic actions, i.e creating and deleting the flows in the slice interacting with the Ryu OpenFlow controller, managing the agents to synchronise the collection of the RTT statistics, requesting the transference of data between the agents, etc.
  - The *Collector* is installed inside a VM in PSNC (server *IBM2*).
- **GUI**: web application developed with AngularJS and composed of four panels (see Fig. 2). The first shows a map with some markups pointing to the locations of the testbeds or partners in EU and JP involved in the slice. The map is also used to highlight the statistics of the links between the DCs. The second is a chart that tracks the time taken for each run of the experiment. The third shows a table containing all available SMOS files distributed in the different *Agents*.

The forth is a simple form used to input the specific file to fetch, the desired target location and the credentials and credentials to access it.

  - The *GUI* is installed inside a VM in PSNC (server *IBM2*).
- **Data visualisation**: third-party software provided by ESA for the SMOS mission, called "*SMOSView*". The software is developed in Java using the Eclipse RCP platform and enables the users to decode and display data from the SMOS data files obtained from the DCs, displaying the contents as images or graphs and exporting the data to a number of alternative formats.
  - The *SMOS Viewer* is installed inside a VM in PSNC (server *IBM2*).
- **Ryu controller**: third-party SDN controller using specific application and some custom extensions to interoperate with the *GUI*.
  - The *Ryu controller* is installed in a server of the PSNC DC.

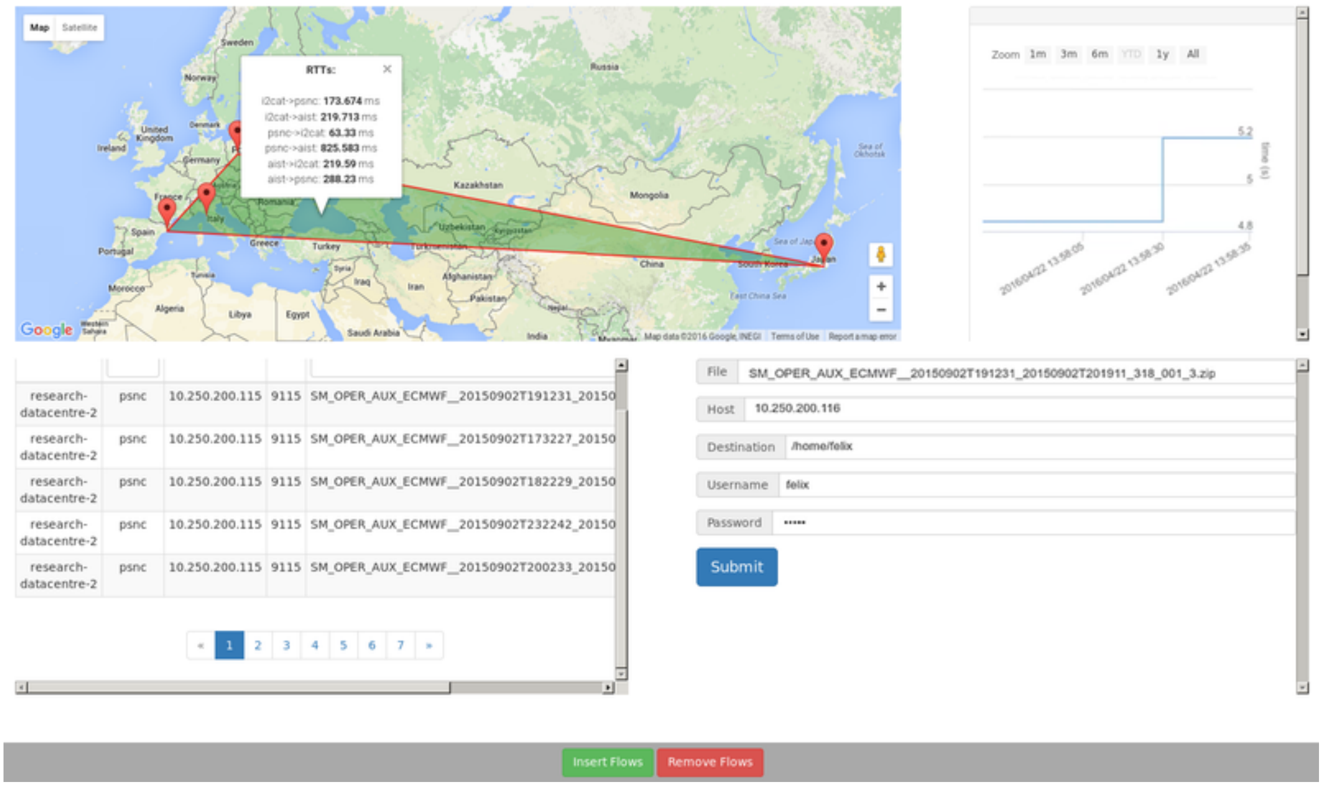
It is worth noting that we enhanced the Ryu controller that manages the slice with some minor extensions:

- Two new methods in the northbound interface of the controller allow to enable and disable (at runtime) other applications.
- The learning switch application can now receive events to enable and disable the sending of the messages.
- The idle-timeout and hard-timeout have been explicitly defined to be natural numbers, i.e. nonzero -- in order to prevent the insertion of permanent rules.

The source code for the Agent and Collector modules, as well as the instructions to install SMOS viewer are available under a specific repository within the FELIX GitHub account (https://github.com/ict-felix/data-on-demand-uc). On the other hand, the demonstration video of this experiment is available under the FELIX channel at YouTube [FELIXTUBE].

3.3.2 Experiment Execution

In this experiment, the end user starts with an environment that provides experimental data within DCs located in i2CAT, PSNC and AIST. The user requests a slice with virtual machines (to deploy custom software inside the DCs to fetch, collect and visualise the data) and with network resources, to interconnect the machines at the different infrastructures. After provisioning this virtual environment, the experiment can manually trigger the set up and tear down of the connections in order to start transmission of a file or to disable or tear down the experiment, respectively. Fig. 2 shows the appearance of the GUI deployed and used by the experiment to communicate with the controller in order to insert or delete flows that provision or delete the connectivity.



RTTs:

i2cat->psnc: **173.674** ms
i2cat->aist: **219.713** ms
psnc->i2cat: **63.33** ms
psnc->aist: **825.583** ms
aist->i2cat: **219.59** ms
aist->psnc: **288.23** ms

research-datacentre-2	psnc	10.250.200.115	9115	SM_OPER_AUX_ECMWF__20150902T191231_20150
research-datacentre-2	psnc	10.250.200.115	9115	SM_OPER_AUX_ECMWF__20150902T173227_20150
research-datacentre-2	psnc	10.250.200.115	9115	SM_OPER_AUX_ECMWF__20150902T182229_20150
research-datacentre-2	psnc	10.250.200.115	9115	SM_OPER_AUX_ECMWF__20150902T232242_20150
research-datacentre-2	psnc	10.250.200.115	9115	SM_OPER_AUX_ECMWF__20150902T200233_20150

File	SM_OPER_AUX_ECMWF__20150902T191231_20150902T201911_318_001_3.zip
Host	10.250.200.116
Destination	/home/felix
Username	felix
Password	•••••

Figure 6: Layout of the collector GUI

As an experimenter, these are the steps to carry in order to run the Use Case:

1. Point the browser to the address where the GUI of the collector module is installed. In this UC we chose "10.216.65.116:9000"
2. Check the polygon area in the map: it will be coloured in red until some flows are installed in the SDN OpenFlow switches and the RTT across domains can be computed
3. Insert the flows by clicking the button, then check again the polygon; until it transitions from red, to yellow, and finally to green
4. In a shell, fetch information from the Ryu controller running at a server in PSNC:
   - curl -X GET http://10.216.65.236:8080/stats/switches (gets list of connected switches)
   - curl -X GET http://10.216.65.236:8080/stats/flow/9354246419888 (gets list of flow entries configured in the SDN device)
5. Now that the polygon is green, all rules are in place and all RTTs will be available. The agents are ready to move the data files between the different DCs
6. Reload the page to check the list of files made available by the agents (shown at the table in the lower, left part of the GUI)
7. Choose one of the files and start the transference/copy from the agents to the SMOS server (here, "10.250.200.116"); using the form located at the lower, right part of the GUI
8. Access the destination folder on the SMOS server (here, "/home/felix" at "10.250.200.116"). The SMOS data files requested will appear here
9. Show the chart in the GUI that summarises the time required for the transference
10. Tear-down the SDN, local links, by removing the flows through the specific button

11. Check again information from the controller and in the polygon. When the rules are all removed, all RTTs should be zero again and the controller should advertise zero flows from the switches

3.3.3 Analysis of the Results

In this section we recall the table with the KPIs presented in the D4.2 and we introduce a new column that describes its status (i.e achieved or not) and a brief description.

Specific Goal / Criteria	FELIX Approach	Status
Resource composition	The FELIX slice allocation, provisioning or perform operational action over the resources should be directly managed by the FELIX management framework	Done. The operations are sent to an MRO, which propagates them to another MROs, ROs or RMs; according to which manages the required resources
Slice creation / modification / destruction time	The FELIX framework allows to automating and reducing time to set up and tear down resources over the different islands	Done. When all resources are set up in the slice, a single operation is enough to set up, manage and tear down resources located in different RMs and islands
Interconnectivity between islands using NSI	FELIX framework will be able to define links over combined SDN and NSI networks	Done. The Use Case provisions through FMS one link between PSNC and AIST using the NSI Connection Service
Provide the user with a global view of the topology the FELIX infrastructure	FELIX framework can discover the complete topology and provide the user with it, so as to set up the environment	Done. The monitoring system provides the graphical representation of both the physical and slice topologies
Determine minimum delay between satellite and research DCs	The FELIX framework will provide easy operations to determine availability of the ROs and RMs in order for the user to identify the delay	Done. Custom monitoring process is used within the collector to retrieve RTTs to each other location or island
Perform data transfers inside a satellite DC	FELIX framework will allow dynamically setting up the SDN paths to transfer the experimenting data between machines	Done. Intra-domain connectivity within the SDN data plane has been achieved, yet transfer rates have been noticed to

	located within a single domain (satellite DC)	be low
Allow traffic between heterogeneous domains	The FELIX framework will take care of identifying and forwarding traffic from SDN domain to the transit network domain and vice-versa	Done. SDN traffic at any given domain moves through the SE device, where a VLAN translation is performed to enter the TN domain, disregarding its type (NSI or GRE)
Perform data transfers between dispersed domains, according to size of data and infrastructure requirements	The FELIX framework allows requesting links through different types of transit networks (NSI, etc.) to interconnect dispersed domains (satellite-to-research DCs)	Done. The FELIX framework sets up the appropriate inter-domain links using GRE and NSI to allow transmission of data across domains

Table 6: KPI UC Data Pre-Processing On Demand

3.3.4 Conclusions

The Data Domain Use Case has proven that it is possible for end users to dynamically orchestrate connectivity between resources through R&D networks and infrastructures that deployed the FELIX Management Stack. In this aspect, FMS is suited for orchestrating the network in a way that both the inter-domain and transit domain details can be completely hidden. On the other hand, the generation of VMs by FMS does not automate configuration details for the set up of the experiment VLAN; thus this is left to the hosting server (where no further configuration is required by the user in the machine, but it restricts to an specific VLANs) or to the experimenter (with freedom of choice for VLANs, but required an extra configuration step).

From the point of view of the user, an initial step is required to clearly define the experiment and its needs and translate it to the data model used in FMS. Once that is achieved, requesting, provisioning and activating resources is a straightforward procedure. The downsides are related to the learning curve required to define the experiment and configure the resources from the different infrastructures, although the former is eased by the jFed GUI tool and the abstraction performed by the M/RO component. Some more considerations on found issues and possible improvements are included as part of the appendix.

3.4 UC. OPOSSUM

3.4.1 Description Final Deployment

The FELIX project offers EU-Japan infrastructure resources for Salzburg Research team in order to perform an OPOSSUM project [OPOSSUM] use-case related to handling of critical traffic by SDN infrastructure. The goal of the OPOSSUM experiment is to measure critical traffic re-route times over long-distance SDN network links in case of detection of the link failure. In order to perform this experiment, FELIX decided to offer access to FELIX islands in PSNC, iMinds and KDDI as well to establish NSI links between those island for purpose of the experiment. However, during more detailed discussions about experiment requirements, we found that the experiment requires usage of OpenFlow 1.3 versions for SDN switches whereas FELIX Software Stack is enabled only for OpenFlow 1.0 (it is dependency of ONL FlowVisor tool which is hard to be replaced by other tool). The approach for execution of the OPOSSUM experiment was changed and FELIX project decided to offer pure infrastructure resources within PSNC and KDDI islands. On Fig. 61, it is presented the experiment topology which is under setup during the time when this document was written.

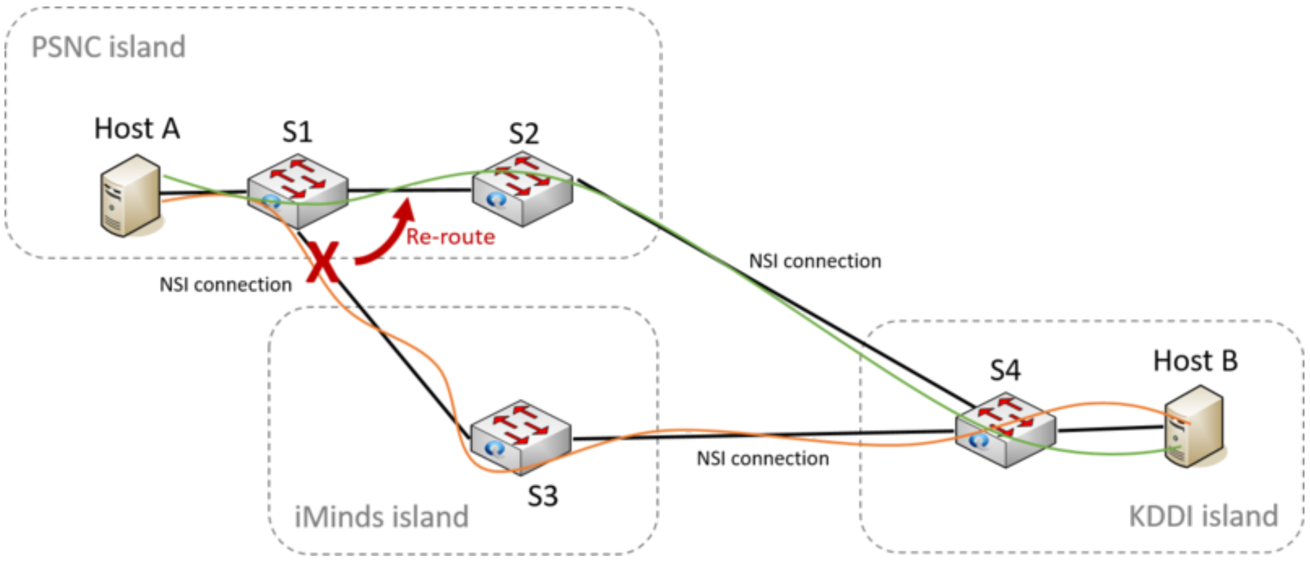


Figure 7: Critical traffic re-route use-case by the OPOSSUM project

3.4.2 Experiment Execution

During writing time of this document, the deployment of the experiment is still in progress and experiment is planned to be executed in the near future.

3.4.3 Analysis of the Results

Still executing during writing time of this document.

3.4.4 Conclusions

Still executing during writing time of this document.

4 Conclusions

Three different use cases were successfully implemented during the FELIX project to validate a great number of features provided by the FELIX Management Stack and the FELIX physical infrastructure.

Firstly, the use cases chosen to assess the outcomes of the project (namely "Data Pre-processing on Demand", "High Quality Media Transmission over Long-Distance Networks", "Infrastructure Domain") allowed to perform some interesting experiments over distributed SDN-enabled networks, where slice resources were successfully requested on demand. On the other hand, the implementation of the FELIX use cases gave some useful feedback to the developers of the FELIX Management Stack. The collected information allowed to improve software components of the stack (e.g. RO-SERM communication related to static links, improvements in M/RO internal logic and mapping of resources, etc). The execution of those experiments highlighted also some transit network constraints, such as the lack of QoS guarantees and stability issues on the NSI infrastructure. The lessons learned and feedback received during the implementation of the FELIX Use Cases acted as a trigger for implementing new methods of inter-domain network connections based on GRE tunnels, as well as extending support in SERM-related data models in SERM and M/RO, and adding new levels of granularity and simplicity in data models accepted by M/RO.

Executing the "High Quality Media Transmission" experiment at the beginning, earlier than other Use Cases, was a good decision; as such Use Case was less dependent on FMS (compared to the other UCs) and thus had much less requirements. This UC also acted as an important trigger to start stressing and finally enabling FELIX infrastructure and inter-domain connections (NSI connections between EU and JP started to work soon after this experiment ended).

Results from the implementation of the FELIX Use Cases were useful also for NSI community and their development team. During the FELIX experiments, we collected some performance measures that may be reused in the future publications or during next experiments; like the estimation of the network capacity needed for the transmission of the media, time of live VM migration and such. Also, some additional software components were developed (e.g. "*RateLimiter*" module, monitoring systems, network applications, collecting agents, GUIs) and they are available on the public FELIX GitHub organization account under their respective repositories.

When preparing and executing experiments, FELIX team members acted as users of the FELIX federated infrastructure services, and used user tools like OMNI CLI and jFed GUI; either requesting to GENIv3-enabled MRO and RO components but also directly to various Resource Managers in order to make minor changes or incremental additions to the experiment.

The FELIX infrastructure was also used by external experimenters (i.e., the "Salzburg Research" team working in the OPOSSUM project). This experiment also brought some clear feedback about the appreciation of the support for newer versions of OpenFlow by the experimenters; where a reimplementation of the SDNRM to an up-to-date version of OpenFlow is much desired.

5 References

[D2.1]	http://www.ict-felix.eu/wp-content/uploads/2014/04/FELIX-D2.1.pdf , online, available 28.04.2015.
[D2.2]	https://wiki.man.poznan.pl/felix/img_auth.php/5/5e/FELIX_D2.2_General_Architecture_and_Functional_Blocks.pdf , online, available 11.02.2014
[Qureshi, Asfandyar et al., 2009]	Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. 2009. Cutting the electric bill for internet-scale systems. *SIGCOMM Comput. Commun. Rev.* 39, 4 (August 2009), 123-134. DOI=10.1145/1594977.1592584.
[OPOSSUM]	The OPOSSUM project, http://www.salzburgresearch.at/en/projekt/opossum-2/
[FELIXGHUB]	FELIX GITHUB, https://github.com/ict-felix/high-quality-media-uc
[FELIXTUBE]	YouTube FELIX's, https://www.youtube.com/watch?v=hScVYgMmuLE

6 Acronyms

API	Application Programming Interface
CBAS	Certificate-based AAA for SDN Experimental Facilities
CLI	Command-Line Interface
CRM	Compute and storage Resource Manager
DC	Data Center
FIRE	Future Internet Research & Experimentation
FMS	FELIX Management Stack
GLIF	Global Lambda Integrated Facility
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
HaaS	Hardware as a Service
IaaS	Infrastructure as a Service
IT	Information Technology
JFED	Java-based framework for testbed federation
KPI	Key Performance Indicator
KVM	Kernel-based Virtual Machine
MS	Monitoring System
M/CBAS	Master-CBAS
M/RO	Master-RO
M/MS	Master-MS
NSI	Network Service Interface
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
RM	Resource Manager
RO	Resource Orchestrator
SDN	Software Defined Networking
SDNRM	SDN Resource Manager
SERM	Stitching Entity Resource Manager
STP	Service Termination Point
TN	Transit Network
TNRM	Transit Network Resource Manager
UC	Use Case
VLAN	Virtual Local Area Network
VM	Virtual Machine

VPN	Virtual Private Network

7 Appendix: UC Implementation Guides

7.1 Guide Data Pre-processing on Demand

7.1.1 Resources and Components Required

RSpec for creating the experimental slice:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
    xs:schemaLocation="http://www.geni.net/resources/rspec/3
                http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
                http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
                http://www.geni.net/resources/rspec/3/request.xsd"
    xmlns="http://www.geni.net/resources/rspec/3"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
    xmlns:felix="http://ict-felix.eu/serm_request"
    xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
    xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
    xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">

  <!-- C @ i2CAT -->
  <node client_id="UCi2CATNXWVerdaguer"
    component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Verdaguer"
    component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm"
    exclusive="true">
   <sliver_type name="emulab-xen">
    <emulab:xen cores="1" ram="512" disk="8"/>
    <disk_image name="urn:publicid:IDN+wall2.ilabt.iminds.be+image+emulab-ops//DEB60_64-VLAN"/>
   </sliver_type>
  </node>
  <node client_id="UCi2CATNXWRodoreda"
    component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Rodoreda"
    component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm"
    exclusive="true">
   <sliver_type name="emulab-xen">
    <emulab:xen cores="1" ram="512" disk="8"/>
    <disk_image name="urn:publicid:IDN+wall2.ilabt.iminds.be+image+emulab-ops//DEB60_64-VLAN"/>
    </sliver_type>
  </node>

  <!-- C @ PSNC -->
  <node client_id="UCi2CATNXWBlade3"
       component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-blade-3"
       component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
       exclusive="true">
     <sliver_type name="emulab-xen">
       <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
       <emulab:xen cores="1" ram="512" disk="10"/>
       <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
     </sliver_type>
  </node>
  <node client_id="UCi2CATNXWIBM2"
       component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-ibm2"
       component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
       exclusive="true">
     <sliver_type name="emulab-xen">
       <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
       <emulab:xen cores="1" ram="512" disk="10"/>
       <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
     </sliver_type>
  </node>

  <!-- C @ AIST -->
  <node client_id="UCi2CATNXWDC14"
       component_id="urn:publicid:IDN+ocf:aist:vtam+node+dc1-4"
       component_manager_id="urn:publicid:IDN+ocf:aist:vtam+authority+cm"
       exclusive="true">
     <sliver_type name="emulab-xen">
       <emulab:xen cores="1" ram="512" disk="10"/>
       <disk_image name="urn:publicid:IDN+wall2.ilabt.iminds.be+image+emulab-ops//DEB60_64-VLAN"/>
     </sliver_type>
  </node>

  <!-- SDN @ i2CAT -->
    <openflow:sliver email="carolina.fernandez@i2cat.net" description="OF-request for i2CAT island">
       <openflow:controller url="tcp:10.216.65.236:6633" type="primary"/>
       <openflow:group name="i2CAT">
          <openflow:datapath
component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:01"
                         component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm"
                         dpid="00:10:00:00:00:00:00:01">
            <openflow:port name="GBE0/3" num="3"/>
            <openflow:port name="GBE0/6" num="6"/>
            <openflow:port name="GBE0/12" num="12"/>
          </openflow:datapath>
          <openflow:datapath
component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:03"
                         component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm"
                         dpid="00:10:00:00:00:00:00:03">
            <openflow:port name="GBE0/1" num="1"/>
            <openflow:port name="GBE0/6" num="6"/>
            <openflow:port name="GBE0/12" num="12"/>
```

```
        </openflow:datapath>
      </openflow:group>
      <openflow:match>
        <openflow:use-group name="i2CAT"/>
        <openflow:packet>
          <openflow:dl_vlan value="2978" />
        </openflow:packet>
      </openflow:match>
    </openflow:sliver>

  <!-- SDN @ AIST -->
  <openflow:sliver email="carolina.fernandez@i2cat.net" description="OF-request for AIST island">
    <openflow:controller url="tcp:10.216.65.236:6633" type="primary"/>
    <openflow:group name="AIST">
      <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:aist:ofam+datapath+00:00:00:00:00:00:00:01"
                    component_manager_id="urn:publicid:IDN+openflow:ocf:aist:ofam+authority+cm"
                    dpid="00:00:00:00:00:00:00:01">
        <openflow:port num="5" name="eth3"/>
        <openflow:port num="6" name="eth4"/>
        <openflow:port num="7" name="eth6"/>
      </openflow:datapath>
    </openflow:group>
    <openflow:match>
      <openflow:use-group name="AIST"/>
      <openflow:packet>
        <openflow:dl_vlan value="1795" />
      </openflow:packet>
    </openflow:match>
  </openflow:sliver>

  <!-- SDN @ PSNC -->
  <openflow:sliver email="carolina.fernandez@i2cat.net" description="OF-request for PSNC island">
    <openflow:controller url="tcp:10.216.65.236:6633" type="primary"/>
    <openflow:group name="PSNC">
      <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+datapath+00:00:08:81:f4:88:f5:b0"
                    component_manager_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+authority+cm"
                    dpid="00:00:08:81:f4:88:f5:b0">
        <openflow:port name="ge-1/1/1.0" num="11"/>
        <openflow:port name="ge-1/1/4.0" num="14"/>
        <openflow:port name="ge-1/1/6.0" num="16"/>
        <openflow:port name="ge-1/1/7.0" num="17"/>
      </openflow:datapath>
    </openflow:group>
    <openflow:match>
      <openflow:use-group name="PSNC"/>
      <openflow:packet>
        <openflow:dl_vlan value="3000" />
      </openflow:packet>
    </openflow:match>
  </openflow:sliver>

  <!-- Virtual link - GRE [ i2CAT - PSNC ] -->
  <link client_id="urn:publicid:IDN+fms:i2cat:mapper+link+i2cat_psnc">
    <component_manager name="urn:publicid:IDN+fms:i2cat:mapper+authority+cm"/>
    <link_type name="urn:felix+virtual_link+type+gre"/>
```

```
   <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+i2cat"/>
   <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+psnc"/>
  </link>

  <!-- Virtual link - NSI [ PSNC - AIST ] -->
  <link client_id="urn:publicid:IDN+fms:psnc:mapper+link+psnc_aist">
   <component_manager name="urn:publicid:IDN+fms:psnc:mapper+authority+cm"/>
   <link_type name="urn:felix+virtual_link+type+nsi"/>
   <interface_ref client_id="urn:publicid:IDN+fms:psnc:mapper+domain+psnc"/>
   <interface_ref client_id="urn:publicid:IDN+fms:psnc:mapper+domain+aist"/>
  </link>

  <!-- Virtual link - GRE [ i2CAT - AIST ] -->
  <link client_id="urn:publicid:IDN+fms:i2cat:mapper+link+i2cat_aist">
   <component_manager name="urn:publicid:IDN+fms:i2cat:mapper+authority+cm"/>
   <link_type name="urn:felix+virtual_link+type+gre"/>
   <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+i2cat"/>
   <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+aist"/>
  </link>

</rspec>
```

Hardware

The slice is composed of 5 VMs (C resources), 4 OpenFlow switches (SDN resources), 3 stitching entity devices (SE resources) and 3 GRE hosts (GRE-TN resources). The resources are located in 3 different islands of the FELIX testbed (i2CAT, PSNC and AIST) interconnected through GRE tunnels (AIST - i2CAT and i2CAT - PSNC) and NSI link (AIST - PSNC).

Software

The entire FELIX Management Stack has been used to allocate the slice, i.e the Master ROs, island ROs, TNRM (GRE and NSI), SERM, SDNRM and CRM; as well as CBAS for the initial generation of the credentials for the slice. Moreover, software has been developed to properly manage the use case life-cycle, as briefly summarised in the section *Merge UC - Data Pre-processing on Demand* of this document. In short, we have implemented 3 modules from scratch, i.e. the agents, the collector and the GUI. A third-party software, the SmosViewer, has been integrated in the slice in order to have a graphical representation of the SMOS data files.

7.1.2 Step by Step Setup

The storyline for this Use Case has been divided in three stages: the first one creates the slice using the FELIX Management Stack, the second consists of the transference of the SMOS data files from the agents to the SMOS server and the last stage corresponds to the visualisation of the data through the SMOS visualiser.

First stage

Initially, the required resources are identified and defined in the RSpec, and the slice is created:

1. Identify the slices to be created, e.g. one for the SDN controller, another for the VMs, SDN local connectivity and inter-domain connections

- **Note**: *the experiment can be composed of one or more slices, depending on the granularity required. For experiments lasting in time it is recommended to separate the network requests from that concerning machines. This is so because the network is more dynamic that the computing or storage nodes, and it may be necessary to request some paths (e.g. NSI) right before the experiment starts. Separating in multiple slices is also a good practice to compartmentalise risks due to eventual connectivity loses or power failures from best-effort infrastructures*

2. Create the slices and generate the credentials for them by providing a specific user credential, the name for the slice, its expected expiration date, and finally the target path to save the newly generated credential file for the slices
   - With omni: `python /opt/geni/geni-tools/src/omni.py -f my_cbas -V3 createslice "dataDomainUC" "2017-01-01T00:00:00+01:00" --cred ~/.gcf/omni-cred.xml --debug -o --slicecredfile ~/.gcf/dataDomainUC -cred.json`
   - The CBAS admin tool allows creating a slice through its GUI, and then fetching the credential through command-line interface:
   - The jFed GUI abstracts the creation of the slice and its credentials; which is done simply by pressing the "Run" experiment button
3. Compose the RSpec to request the resources and submit to any MRO (requests will be distributed to the appropriate MRO or RO). In this UC we pointed to the MRO at Europe
   1. Define the RSpec to request the SDN controller
      - **Note**: *the SDN Ryu controller was deployed directly in a server in the PSNC DC, thus we did not longer require to create a machine for it*
   2. Define the RSpec to request resources with virtual machines, SDN flowspaces and two GRE connections and one NSI connection between the i2CAT, PSNC and AIST islands
   - **Note 1**: *the SDN flowspace must point to the IP of the device where the Ryu controller is deployed (see step above or check manifest for IP of the VM after creating a new one)*
   - **Note 2**: *the RSpec for this Use Case relies on virtual links, which generate and abstract inter-domain connections (transit VLANs, STPs, ...) between different islands so that the experimenter does not care about it. However, NSI connection is more reliable when using specific VLAN; and the request procedure may fail. In this case the specific NSI details should be explicitly provided in the RSpec*
4. Use any client (jFed or omni) to allocate, provision and start the resources to the M/RO
   - With omni:
     - Allocate: `python /opt/geni/geni-tools/src/omni.py -f my_cbas -V3 -a https://127.0.0.1:18440/xmlrpc/geni/3/ allocate dataDomainUC ~/omni/use_case/uc__full_vls.xml --slicecredfile ~/.gcf/dataDomainUC-cred.json`

  - Provision: `python /opt/geni/geni-tools/src/omni.py -f my_cbas -V3 -a https://127.0.0.1:18440/xmlrpc/geni/3/ provision dataDomainUC --slicecredfile ~/.gcf/dataDomainUC-cred.json`
  - Start resources: `python /opt/geni/geni-tools/src/omni.py -f my_cbas -V3 -a https://127.0.0.1:18440/xmlrpc/geni/3/ poa dataDomainUC geni_start --slicecredfile ~/.gcf/dataDomainUC-cred.json`
- With jFed: create a new experiment, move to the RSpec editor tab and paste it, then press the "Run" button and set "dataDomain" as the name of the slice

After setting up the slice, we must proceed to the configuration of the virtual machines in order to configure the network and package repositories to continue installing the required software (e.g. agents, collectors). The following steps account for the configuration of the VMs:

1. Fix any connectivity or DNS-related problem:
   - `echo "nameserver 8.8.8.8" >> /etc/resolv.conf`
2. Fix any problem related to */etc/apt/sources.list* by updating with proper, up-to-date sources
3. Check the routing tables to ensure everything is properly set:
   - In the terminal of the VMs at i2CAT.Rodoreda and i2CAT.Verdaguer: `route add -net 10.250.200.0 netmask 255.255.255.0 dev eth1.2978 && ip route delete 10.0.0.0/8 dev eth1.2978`
   - In the terminal of the VM at PSNC.IBM2, remove any duplicate for entry *10.0.0.0*
4. Add private IPs (in the range *10.250.xx.xx/16*, configured for NSI) to each VM; so that data transmission can flow through the SDN data plane
   - **Note**: *the configuration of each VM varies, as one or another interface may be used; and also due to differences in the VLAN tagging, at some points performed in the server providing the VMs and at other points configured manually by the user*
   - In the terminal of the VM at i2CAT.Rodoreda: ifconfig eth1.2978 10.250.200.26
   - In the terminal of the VM at i2CAT.Verdaguer: ifconfig eth1.2978 10.250.200.27
   - In the terminal of the VM at PSNC.Blade3: ifconfig eth1 10.250.200.115
   - In the terminal of the VM at PSNC.IBM2: vconfig add eth1 3000 && ifconfig eth1.3000 up && ifconfig eth1.3000 10.250.200.116
   - In the terminal of the VM at AIST.DC1-4: ifconfig eth1 10.250.200.205

At the end of the first stage, the connectivity configuration must be successfully set in the VMs, as indicated by Fig. 8:

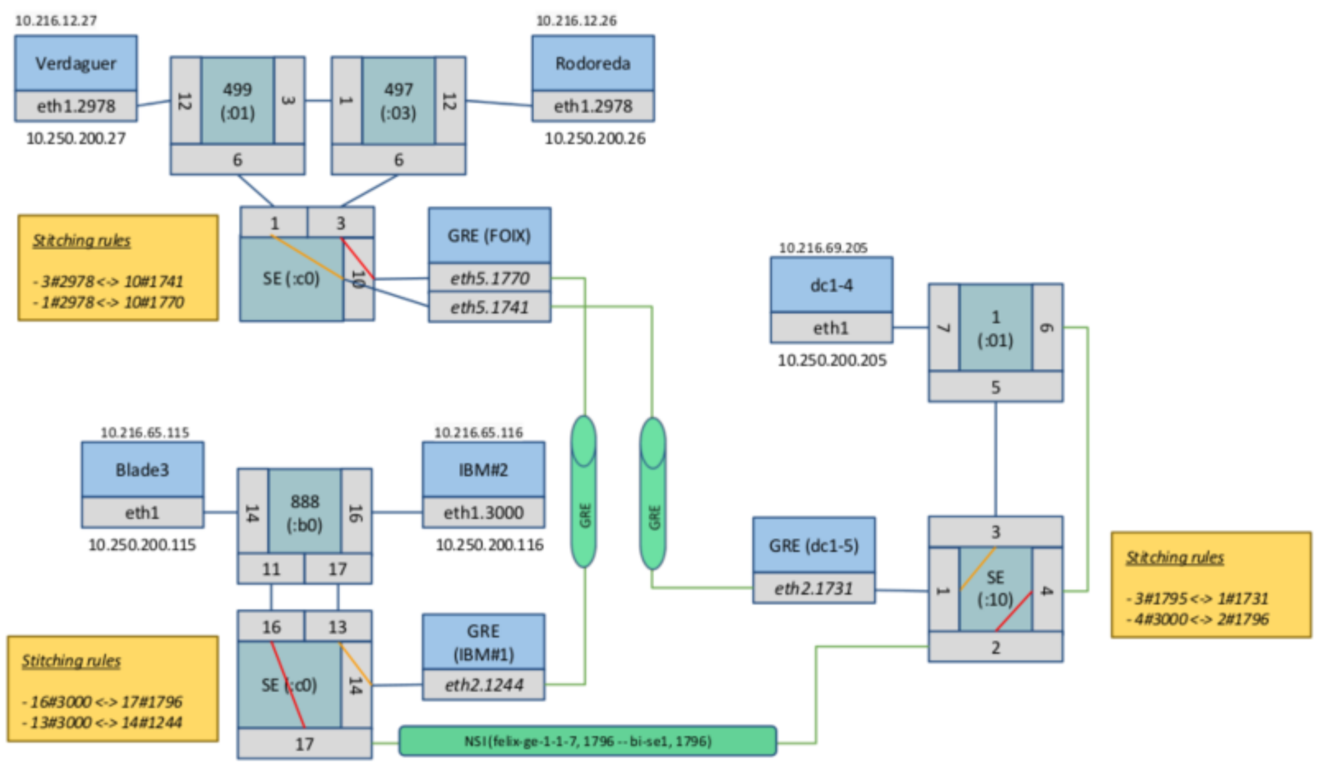


Figure 8: Details on the networks of the slice for the Data pre-processing on demand UC

After this, the custom software necessary for the Use Case shall be installed prior to initiate the second stage.

Second stage

After creating the slice, placing all the data files and configuring the machines, the following step is to trigger transmission of the data files through different domains at FELIX:

1. Point the browser to the address where the GUI of the collector module is installed (10.216.65.116:9000).
2. Click on the polygon zone: the area should be coloured in RED as the RTT values are all zeros as no flow entries are installed in the SDN OpenFlow switches.
3. Click on the "Insert Flows" button. You should receive an alert "Insert Flows success!" (click OK).
4. Click again in the polygon zone. It should be YELLOW now, as some values are still zero.

In fact, the procedure to retrieve the RTT values is a time-based thread implemented in the collector server and takes few minutes to complete.

1. Open a shell and try to retrieve some info from the Ryu controller running at a server in PSNC, e.g.:
   - `curl -X GET http://10.216.65.236:8080/stats/switches (to get the list of connected switches)`
   - `curl -X GET http://10.216.65.236:8080/stats/flow/9354246419888 (to get the list of the flow entries configured in the device)`
   - **Note**: *the Ryu controller has been enhanced with minor extensions and configurations:*

- *Two new methods in the northbound interface of the controller allow enabling/disabling (at runtime) other applications*
- *The learning switch application has been enhanced to receive events to enable and disable the sending of the messages*
- *The idle-timeout and the hard-timeout have been fixed to not install permanent rules*

2. Click again in the zone and wait to have a GREEN. When this happens, the agents are ready to move the data files.
3. At this point, just reload the page and see the files that are available in the agents in the table of the GUI.
4. Start copying files from the agents to the SMOS server (10.250.200.116) using the proper form and the *Submit* button. For example you can compile the form using the following values:
   - *File: SM_OPER_AUX_ECMWF__20150902T191231_20150902T201911_318_001_3.zip*
   - *Host: 10.250.200.116*
   - *Destination: /home/felix*
   - *Username: felix*
   - *Password (password field): felix*
5. Show the destination folder of the SMOS server (/home/felix at the address 10.250.200.116). At the beginning the folder is empty, but after each operation increases with the SMOS data files.
   - **Note**: *at the end of each operation, the GUI shows the time spent to do the action in a dedicated stock-chart. Note that we can have long times to complete the operations as we are moving big files in GRE tunnels, in some cases.*
6. Show the chart in the GUI that summarises the results.
7. Tear-down the links removing the flows. Click on the "Remove Flows" button. You should receive an alert "Remove Flows success!" (click OK).
8. Open a shell and show the results of the curl commands (zero length lists). The RTT values should became zeros in the GUI progressively.

It is important to note that there are 2 agents in the slice that manage exactly the same files, located at PSNC and i2CAT. The collector should select the closest agent to start the procedure and use the RTT values to take decisions Basically the procedure can be summarised as:

1. The collector receives the request to retrieve a data file from the experimenter
2. The collector localises the owner of the file through proper messages
3. If more than one agent has a copy of the data file, the agent is chosen by the minimum RTT value
4. A request to the selected agent is forwarded.

Third stage

After retrieving the data files from specific DCs, the final steps are focused on visualising them:

1. Install the SMOS viewer, either at the DC or locally
   - **Note**: *the guide "howto_install_smosview" describes the steps to install the application*
2. Run the SMOS view graphical interface in the server where it is installed

3. Through CLI, type: ./smosview
    - **Note**: *it was problematic to run this application through a "double" VPN channel: the GUI could not be displayed with the X-forwarding enabled in the ssh tunnel. After trying different SSH parameters and less computation-heavy ciphers; we ended up connecting to the collector via a gateway in PSNC to provide local, faster access to the application. Installing the viewer locally is another option*
4. Move in the directory where the "DBL" files are located.
5. Open the "DBL" file (double-click) and click on the satellite icon of the main menu.
6. Select a field using the drop-down menu, e.g. *Air_Temperature_2m*
    - **Note**: *this depends on the content of the SMOS data file*
7. The GUI starts loading the data for the selected field and creating points in the map.

7.1.3 Issues Found

Throughout the deployment of this Use Case we came across a number of technical problems to realise connectivity:

- As a best effort service, the infrastructures may have connectivity problems or power failures from time to time. To address with those problems, we thoroughly documented the configuration steps for the slices, for each VM and the layout of the final environment across the three islands
- Provisioning NSI connectivity has not always been reliable. Using the virtual links mapper proved to be problematic in some cases, as it picks random VLANs and STPs from the pool of available values; and then some circuit provisioning may fail. In those cases, it worked better for us to define explicit STPs and VLANs to request the NSI path between PSNC and AIST islands, along with the route to stitch the SE device
- There were some misunderstandings on how the VLAN tagging worked in the different infrastructures; which misled us to think network connectivity was not achieved during some time. This was due to the different way in which infrastructure tags packets coming out from the VM of the user, i.e. the VLAN tagging is performed in the hosting server at "AIST.dc1-4" and "PSNC.Blade3", each restricting tagging to a specific VLAN; or the VLAN tagging is directly performed by the user at the VM, as in the cases of "PSNC.IBM2", "i2CAT.Rodoreda" and "i2CAT.Verdaguer. It is important to have this in mind when configuring the connectivity of the VMs during the first stage
- The custom application used for this UC on top of the Ryu SDN controller faced problems when inserting flows on some of the physical SDN-enabled devices (i.e. wrong timeout values and unknown buffer id). In the end we used a common L2 learning switch for it, but this may have led to the insertion of some improper flows, e.g. not required for the minimal paths between the islands
- One of the recurring processes running in the background of the collector reported initially very high RTT values between some islands, but in just one of the two directions. This process runs a "ping" to send a single ICMP packet between islands ("ping -c 1") at each end of the links. We believe that due to some initial negotiation or some invalid set-up of the flows, the initial ping reported very high values. An average of the time required to receive back a number of ICMP echo replies could be a better indicative
- The SMOS viewer was deployed on one of the VMs requested, being accessible through the VPN only. When connecting remotely to this machine through X11 and loading the SMOS viewer, it was sometimes not possible to load the application. To overcome this, we ended up with two solutions: i) connecting to this viewer through a gateway machine with a public IP that is located in PSNC, and ii) install remotely and visualising the files there.

7.1.4 Comments

After taking the role of experimenter ourselves, we consider there were some difficult steps more complex than what they should ideally be. For instance:

- The way of tagging VLANs should be ideally completely hidden to the user, and any interface from the VMs would be available for outgoing connectivity. At least, the process of tagging VLANs should be done in an homogeneous way across all the FELIX-enabled infrastructures
- The heterogeneous behaviour of provisioning the network through the NSI Connection Service has limited for our scenario the benefits of the abstraction provided by the virtual links, and may compel the user in some specific scenarios (as in this UC) to perform extra steps and to look for extra details on the STPs and the stitching entity that may not be part of the intended experiment
- Periodic assessment (e.g. automatic) of the connectivity requested by the user would help detecting sooner any incidence due to misconfiguration or power failure within the infrastructure

7.2 Guide Data Mobility

7.2.1 Resources and Components Required

Hardware

- The slice is composed of 2 VMs (C resources), 2 openflow switches (SDN resources), 2 stitching entity devices (SE resources) and 1 Transport Network (NSI-TN resrouces)
- Creating VM must support nested virtualization using KVM. 3 child VMs running on that VM .
- The resources are located in 2 islands of the FELIX testbed (KDDI and PSNC) interconnected through NSI link (KDDI - PSNC).
- 1 client PC, laptop PC(Dynabook R731) with Ubuntu1204
- 1 VM for KDDI SDN controller, 1 core, 1MB memory, Ubuntu1404

Software

The entire FELIX Management Stack was used to allocate the slice. Also, the following software were used for the Data Mobility Service.

- **OpenVPN**
  - Popular software for VPN.
  - VPN is used so that the user can securely connect to their VM (desktop) in the cloud.
- **perfSONAR**
  - Network measurement software.

    - This software is used to measure the delay between the clouds and user client. This data is used to determine the appropriate cloud for the user.
- **VyOS**
    - Software router.
    - This is the border router between the internet and intranet.
- **OpenVSwitch(OVS)**
    - Software Openflow switch.
    - Facilitates changing the traffic flow to the client.
- **KDDI VPN client**
    - Proprietary VPN client developed by KDDI.
    - It clients connect to the cloud with VPN and interacts with the KDDI SDN controller.
- **KDDI SDN controller**
    - Proprietary SDN controller for Data Mobility Service.
    - This orchestrates the VM migration and traffic rerouting.

Network Diagram

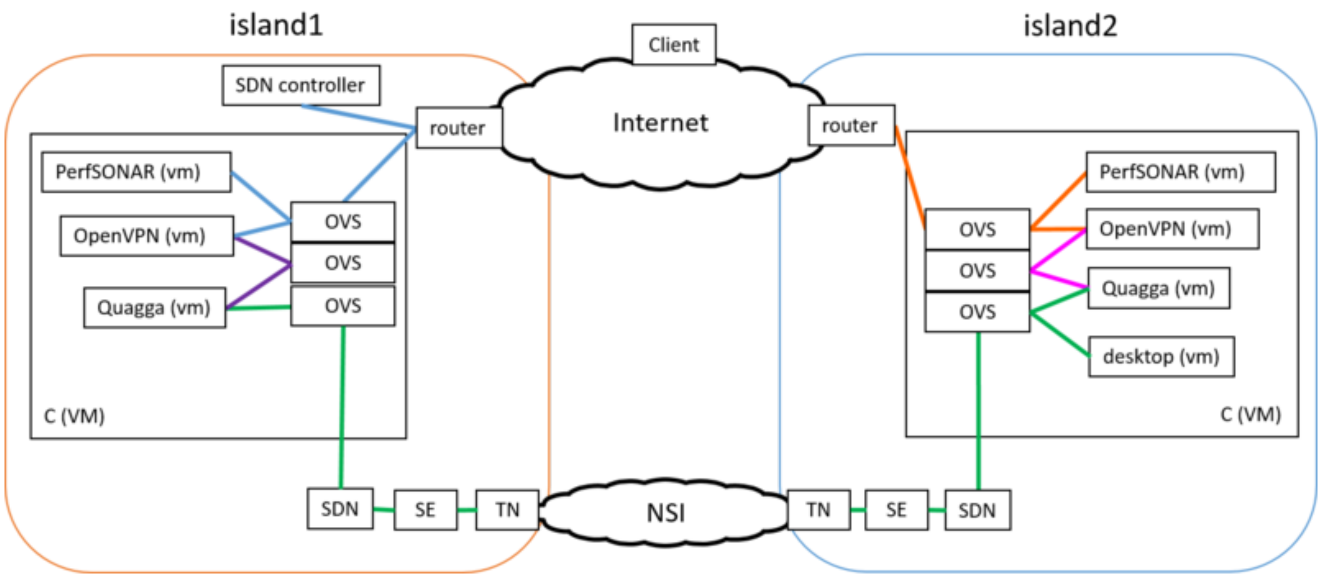


Figure 9: Detail Network Diagram for Data Mobility Service

7.2.2 Step by Step Setup

1. **Slice**

Creating the slice with OMNI

- python omni.py -f my_cbas -a https://<MRO>:18440/xmlrpc/geni/3/ -V3 allocate UCSlice UCSlice.rspec --slicecredfile ~/.gcf/UCSliceCred.json
- python omni.py -f my_cbas -a https://<MRO>:18440/xmlrpc/geni/3/ -V3 provision UCSlice --slicecredfile ~/.gcf/UCSliceCred.json
- python omni.py -f my_cbas -a https://<MRO>:18440/xmlrpc/geni/3/ -V3 poa UCSlice geni_start --slicecredfile ~/.gcf/UCSliceCred.json

RSPEC:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
      xs:schemaLocation="http://www.geni.net/resources/rspec/3
 http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
             http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
       http://www.geni.net/resources/rspec/3/request.xsd"
       xmlns="http://www.geni.net/resources/rspec/3"
       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
       xmlns:felix="http://ict-felix.eu/serm_request"
       xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
       xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
       xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">

 <!-- C@KDDI -->
 <node client_id="kddikvm1"
    component_id="urn:publicid:IDN+ocf:kddi:vtam+node+kddi-cp1"
    component_manager_id="urn:publicid:IDN+ocf:kddi:vtam+authority+cm"
    exclusive="true">
    <sliver_type name="emulab-xen">
      <emulab:xen cores="2" ram="4096" disk="128"/>
      <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
    </sliver_type>
 </node>
 <!-- SDN @ KDDI -->
 <openflow:sliver email="ikeda@ote.kddi.com" description="OF request example">
   <openflow:controller url="tcp:10.216.68.9:6633" type="primary"/>
   <openflow:group name="fs1">
      <openflow:datapath   component_id="urn:publicid:IDN+openflow:ocf:kddi:ofam+datapath+00:00:00:25:5c:e6:4f:07"
component_manager_id="urn:publicid:IDN+openflow:ocf:kddi:ofam+authority+cm" dpid="00:00:00:25:5c:e6:4f:07">
         <openflow:port name="GBE0/14" num="14"/>
         <openflow:port name="GBE0/2" num="2"/>
      </openflow:datapath>
   </openflow:group>
   <openflow:match>
      <openflow:use-group name="fs1" />
      <openflow:packet>
        <openflow:dl_vlan value="1792" />
      </openflow:packet>
   </openflow:match>
 </openflow:sliver>

 <!-- SERM @ KDDI -->
```

```
<node                                   client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8"
component_manager_id="urn:publicid:IDN+fms:kddi:serm+authority+cm">
 <interface client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_2">
  <sharedvlan:link_shared_vlan                                                                vlantag="1792"
name="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_2+vlan"/>
 </interface>
 <interface client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_10">
  <sharedvlan:link_shared_vlan                                                                vlantag="1792"
name="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_10+vlan"/>
 </interface>
</node>
<link client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_2_00:00:00:25:5c:e6:4f:d8_10">
 <component_manager name="urn:publicid:IDN+fms:kddi:serm+authority+cm"/>
 <interface_ref client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_2" felix:vlan="1792"/>
 <interface_ref client_id="urn:publicid:IDN+fms:kddi:serm+datapath+00:00:00:25:5c:e6:4f:d8_10" felix:vlan="1792"/>
 <link_type name="urn:felix+vlan_trans"/>
</link>

 <!-- C@PSNC -->
 <node client_id="kddikvm2"
    component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-blade-5"
    component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
    exclusive="true">
    <sliver_type name="emulab-xen">
     <emulab:xen cores="1" ram="512" disk="128"/>
     <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
    </sliver_type>
 </node>
 <!-- SDN@PSNC -->
   <openflow:sliver email="ikeda@ote.kddi.com" description="OF request example">
      <openflow:controller url="tcp:10.216.68.9:6633" type="primary"/>
      <openflow:group name="fs1">
         <openflow:datapath
component_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+datapath+00:00:08:81:f4:88:f5:b0"
component_manager_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+authority+cm" dpid="00:00:08:81:f4:88:f5:b0">
            <openflow:port name="ge-1/1/2.0" num="12"/>
            <openflow:port name="ge-1/1/9.0" num="19"/>
         </openflow:datapath>
      </openflow:group>
      <openflow:match>
         <openflow:use-group name="fs1" />
         <openflow:packet>
           <openflow:dl_vlan value="792" />
         </openflow:packet>
      </openflow:match>
   </openflow:sliver>

 <!-- SE@PSNC -->
 <node                                  client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0"
component_manager_id="urn:publicid:IDN+fms:psnc:serm+authority+cm">
   <interface client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_3">
    <sharedvlan:link_shared_vlan                                                              vlantag="1792"
name="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_3+vlan"/>
   </interface>
   <interface client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_2">
```

```
    <sharedvlan:link_shared_vlan                                                                    vlantag="792"
name="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_2+vlan"/>
   </interface>
  </node>
  <link client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_3_00:00:54:e0:32:cc:a4:c0_2">
   <component_manager name="urn:publicid:IDN+fms:psnc:serm+authority+cm"/>
   <interface_ref client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_3" felix:vlan="1792"/>
   <interface_ref client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_2" felix:vlan="792"/>
   <link_type name="urn:felix+vlan_trans"/>
  </link>

  <!-- TN (PSNC [ ] - KDDI [ stp1 ]) -->
  <node client_id="urn:publicid:IDN+fms:aist:tnrm+stp"
      component_manager_id="urn:publicid:IDN+fms:aist:tnrm+authority+cm">
   <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:jgn-x.jp:2013:topology:bi-felix-kddi-stp2">
    <sharedvlan:link_shared_vlan  name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:jgn-x.jp:2013:topology:bi-
felix-kddi-stp2+vlan" vlantag="1792"/>
   </interface>
   <interface     client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-0-
3">
    <sharedvlan:link_shared_vlan
name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-0-3"
vlantag="1792"/>
   </interface>
  </node>
  <link                         client_id="urn:publicid:IDN+fms:aist:tnrm+link+urn:ogf:network:jgn-x.jp:2013:topology:bi-felix-kddi-
stp2?vlan=1792-urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-0-3?vlan=1792">
   <component_manager name="urn:publicid:IDN+fms:aist:tnrm+authority+cm"/>
   <interface_ref       client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:jgn-x.jp:2013:topology:bi-felix-kddi-
stp2"/>
   <interface_ref  client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-
0-3"/>
   <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:jgn-x.jp:2013:topology:bi-felix-kddi-stp2"
         dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-0-3"
         capacity="1000">
   </property>
   <property   source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-0-
3"
         dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:jgn-x.jp:2013:topology:bi-felix-kddi-stp2"
         capacity="1000">
   </property>
  </link>

</rspec>
```

2. **KVM**

- Installing KVM 1.0 and create the 4 guest VMs (http://www.linux-kvm.org/)
- 3 VMs are with 2 Core, 2MB memory , 8GB HDD, Ubuntu1204
- 1 VMs are with 2 Core, 2MB memory , 8GB HDD, VyOS

- Connecting VMs to the OVS as diagram

3. **OpenVPN**

- Installing OpenVPN 2.2.1 (https://openvpn.net/) on one of the guest VM
- Configure OpenVPN server so that OpenVPN client can connect and launch it.

4 **VyOS**

- Installing VyOS 3.13 (http://vyos.net/wiki/Main_Page)
- configure the VyOS to be the border router between Internet and Intranet.

5 **perfSONAR**

- Installing perfSONAR 3.5.1 (http://www.perfsonar.net/)
- Configure the PingER to ping the client

6 **KDDI VPN Client**

- Installing Ubuntu1204 on the laptop with GUI
- Install the KDDI VPN packages and configure it.

7 **KDDI SDN controller**

- Installing Java SDK 8
- Install the KDDI SDN contoller and configure it
- launch the controller

7.3 Guide Disaster Recovery by Migrating IaaS to a Remote Data Center

7.3.1 Resources and Components Required

Hardware

This use case was performed in two different ways: (1) between AIST1 and AIST2 islands and (2) between AIST1 and PSNC islands using NSI-enabled R&E networks and GRE over the Internet. In each experiment, the following resources were required:

- The slice is composed of 4 VMs (C resources), 2 OpenFlow switches (SDN resources), 2 stitching entity devices (SE resources), NSI-enabled R&E network (NSI-TN resource) and 2 GRE hosts (GRE-TN resources).
- The C/SDN/SE/GRE-TN resources are located in (1) AIST1 and AIST2, and (2) AIST1 and PSNC, interconnected through NSI link (AIST1 - AIST2, AIST1 - (JGN-X - iCAIR - Netherlight) - PSNC) and GRE tunnels (AIST1 - AIST2 and AIST1 - PSNC).
- Provisioned VMs must support nested virtualization. An IaaS management server VM and user VMs are provisioned over the provisioned VMs.

Software

- The entire FELIX Management Stack was used to allocate the slice, i.e the Master-RO, Island-ROs, TNRM (GRE and NSI), SERM, SDNRM and CRM.
- CloudStack as IaaS management software for the UC.
- RYU controller for controlling SDN switches allocated for the slice.
- A set of scripts were developed to perform the migration of an entire IaaS by AIST.

The followings are example rspecs for IaaS migration between AIST1 and PSNC.

- RSpec for creationg the source SDN island (RSpec1)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
  xs:schemaLocation="http://www.geni.net/resources/rspec/3
        http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
        http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
        http://www.geni.net/resources/rspec/3/request.xsd"
  xmlns="http://www.geni.net/resources/rspec/3"
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
  xmlns:felix="http://ict-felix.eu/serm_request"
  xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
  xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
  xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">
 <node client_id="aist1-manager"
  component_id="urn:publicid:IDN+ocf:aist:vtam+node+dc1-1"
  component_manager_id="urn:publicid:IDN+ocf:aist:vtam+authority+cm"
  exclusive="true">
  <sliver_type name="emulab-xen">
   <emulab:xen cores="1" ram="512" disk="128"/>
   <disk_image name="/mnt/l1vm/template/l1vm-mng-20151116.qcow2"/>
  </sliver_type>
 </node>
 <node client_id="aist1-agent1"
  component_id="urn:publicid:IDN+ocf:aist:vtam+node+dc1-2"
  component_manager_id="urn:publicid:IDN+ocf:aist:vtam+authority+cm"
  exclusive="true">
  <sliver_type name="emulab-xen">
   <emulab:xen cores="1" ram="512" disk="128"/>
   <disk_image name="/mnt/l1vm/template/l1vm-agent.qcow2"/>
  </sliver_type>
 </node>
 <openflow:sliver email="atsuko.takefusa@aist.go.jp"
      description="FELIX AIST-PSNC IaaS Migration Demo">
  <openflow:controller url="tcp:10.216.69.137:6633" type="primary"/>
  <openflow:group name="AIST">
   <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:aist:ofam+datapath+00:00:00:00:00:00:00:01"
       component_manager_id="urn:publicid:IDN+openflow:ocf:aist:ofam+authority+cm"
       dpid="00:00:00:00:00:00:00:01">
  <openflow:port num="6" name="eth4"/>
  <openflow:port num="1" name="eth12"/>
  <openflow:port num="2" name="eth13"/>
```

```
    </openflow:datapath>
   </openflow:group>
   <openflow:match>
    <openflow:use-group name="AIST"/>
    <openflow:packet>
   <openflow:dl_vlan value="1795" />
    </openflow:packet>
   </openflow:match>
  </openflow:sliver>
</rspec>
```

- RSpec for creationg the remote SDN island (RSpec2)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
   xs:schemaLocation="http://www.geni.net/resources/rspec/3
           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
           http://www.geni.net/resources/rspec/3/request.xsd"
   xmlns="http://www.geni.net/resources/rspec/3"
   xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
   xmlns:felix="http://ict-felix.eu/serm_request"
   xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
   xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
   xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">
  <node client_id="psnc-manager"
   component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-ibm2"
   component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
   exclusive="true">
   <sliver_type name="emulab-xen">
    <emulab:xen cores="1" ram="512" disk="128"/>
    <disk_image name="/mnt/l1vm/template/l1vm-mng.qcow2"/>
   </sliver_type>
  </node>
  <node client_id="psnc-agent1"
   component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-ibm2"
   component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
   exclusive="true">
   <sliver_type name="emulab-xen">
    <emulab:xen cores="1" ram="512" disk="128"/>
    <disk_image name="/mnt/l1vm/template/l1vm-agent.qcow2"/>
   </sliver_type>
  </node>
  <openflow:sliver email="atsuko.takefusa@aist.go.jp"
        description="FELIX AIST-PSNC IaaS Migration Demo">
   <openflow:controller url="tcp:10.216.69.137:16633" type="primary"/>
   <openflow:group name="AIST">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+datapath+00:00:08:81:f4:88:f5:b0"
         component_manager_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+authority+cm"
         dpid="00:00:08:81:f4:88:f5:b0">
   <openflow:port num="11" name="ge-1/1/1.0"/>
   <openflow:port num="14" name="ge-1/1/4.0"/>
   <openflow:port num="16" name="ge-1/1/6.0"/>
```

```
      </openflow:datapath>
    </openflow:group>
    <openflow:match>
      <openflow:use-group name="AIST"/>
      <openflow:packet>
    <openflow:dl_vlan value="3000" />
      </openflow:packet>
    </openflow:match>
  </openflow:sliver>
</rspec>
```

- RSpec for creationg a path between two island (RSpec3)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
    xs:schemaLocation="http://www.geni.net/resources/rspec/3
            http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
            http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
            http://www.geni.net/resources/rspec/3/request.xsd"
    xmlns="http://www.geni.net/resources/rspec/3"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
    xmlns:felix="http://ict-felix.eu/serm_request"
    xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
    xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
    xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">
  <node client_id="urn:publicid:IDN+fms:aist:tnrm+stp"
    component_manager_id="urn:publicid:IDN+fms:aist:tnrm+authority+cm">
    <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2013:topology:bi-se1">
      <sharedvlan:link_shared_vlan
name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2013:topology:bi-se1+vlan" vlantag="1796"/>
    </interface>
    <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-1-
7">
      <sharedvlan:link_shared_vlan
name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-1-7+vlan"
vlantag="1796"/>
    </interface>
  </node>
  <link client_id="urn:publicid:IDN+fms:aist:tnrm+link+urn:ogf:network:aist.go.jp:2013:topology:bi-se1?vlan=1796-
urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-1-7?vlan=1796">
    <component_manager name="urn:publicid:IDN+fms:aist:tnrm+authority+cm"/>
    <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2013:topology:bi-se1"/>
    <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-
1-7"/>
    <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2013:topology:bi-se1"
        dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-1-7"
        capacity="1000">
    </property>
    <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2013:topology:felix-ge-1-1-
7"
        dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2013:topology:bi-se1"
        capacity="1000">
    </property>
  </link>
  <node client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10"
component_manager_id="urn:publicid:IDN+fms:aist:serm+authority+cm">
     <interface client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_4">
   <sharedvlan:link_shared_vlan vlantag="1795"
name="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_4+vlan"/>
     </interface>
     <interface client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_2">
   <sharedvlan:link_shared_vlan vlantag="1796"
name="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_2+vlan"/>
     </interface>
  </node>
  <link client_id="urn:publicid:IDN+fms:aist:serm+00:00:00:00:00:00:00:10_4_00:00:00:00:00:00:00:10_2">
     <component_manager name="urn:publicid:IDN+fms:aist:serm+authority+serm"/>
     <link_type name="urn:felix+vlan_trans"/>
     <interface_ref client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_4" felix:vlan="1795"/>
     <interface_ref client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_2" felix:vlan="1796"/>
  </link>
  <node client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0"
   component_manager_id="urn:publicid:IDN+fms:psnc:serm+authority+cm">
   <interface
   client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_16">
   <sharedvlan:link_shared_vlan vlantag="3000"
     name="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_16+vlan" />
   </interface>
   <interface
   client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_17">
   <sharedvlan:link_shared_vlan vlantag="1796"
     name="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_17+vlan" />
   </interface>
  </node>
  <link
   client_id="urn:publicid:IDN+fms:psnc:serm+00:00:54:e0:32:cc:a4:c0_16_00:00:54:e0:32:cc:a4:c0_17">
   <component_manager name="urn:publicid:IDN+fms:psnc:serm+authority+serm" />
   <link_type name="urn:felix+vlan_trans" />
   <interface_ref
   client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_16"
   felix:vlan="3000" />
   <interface_ref
   client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_17"
   felix:vlan="1796" />
  </link>
</rspec>
```

7.3.2 Step by Step Setup

1. Checking configuration files before an experiment
   - iaas_param.sh: General parameters
   - iaas_param_aist.sh: Sample settings for AIST1 and AIST2
   - iaas_param_psnc.sh: Sample settings for AIST1 and PSNC

- iaas_ssh.conf: ssh configuration
- iaas_l1domxml.sed: VM launch parameters for L1VM (sed script for set correct parameters)
- iaas_l2domxml.sed: VM launch parameters for L2VM (sed script for set correct parameters)
- Details are described in iaas_README_en.txt in the IaaS migration package.

2. Construction of IaaS in the source SDN island.
   - **$ sh iaas_makeslices.sh**
   - Allocate and provision resources for a slice in the source SDN island (DC1).
   - Setup an IaaS environment, including management servers, computing resources and VMs provisioned on them, using the CloudStack GUI.
   - **$ sh iaas_START_DC1.sh**
   - Launch IaaS environment in DC1 using (RSpec1).
3. Slice extension to the remote SDN island.
   - **$ sh iaas_START_DC2.sh** using (RSpec2) and (RSpec3)
   - Allocate and provision resources in the destination SDN island (DC2) and a transit network (NSI path or GRE tunnel) between DC1 and DC2.
4. Stop and transfer the IaaS environment.
   - Stop running VMs and disable the IaaS environment using the CloudStack GUI.
   - **$ iaas_COPY_L2VM.sh**
   - Make the whole the IaaS snapshot and copy the snapshot to DC2.
5. Completion of IaaS migration.
   - **$ sh iaas_START_L2VM_DC2.sh**
   - Resume the transferred IaaS environment in DC2.
   - Restart VMs using the CloudStack GUI.

7.4 Guide High Quality Media Transmission over Long-Distance Networks

7.4.1 Resources and Components Required

Hardware

The Use Case tried to use 8 OpenFlow switches (SDN resources), 5 stitching switches (SE resources) and two dedicated servers with UltraGrid software (out of scope of CRM component management). The resources were located in five different islands of the FELIX testbed (PSNC, i2CAT, iMinds, KDDI and AIST) interconnected through 4 GRE tunnels (AIST - PSNC) and static links (PSNC - i2cat, i2cat - iMinds), Geant BoD link (PSNC - iMinds). Interconnections using NSI AutoGOLE service for all EU-JP links were unsuccessful which caused that big part of the physical infrastructure were controlled by experimenter OF controller but real traffic was not transported (instead real traffic was send via GRE tunnels).

The following additional hardware was interconnected in PSNC and AIST to perform the experiment:

- Media workstation with Intel i7, 12GB RAM, AMD Fire Pro 3D Graphics v8800 for media player
- Sony SXRD 4K Ultra-high Resolution Projector

Software

The Use Case tried to use SDNRM and SERM components in all five islands but, at the end, SERM was configured manually similarly to SDNRM's flowvisor. Additionally, it was deployed software components dedicated for the Use Case:

- UtraGrid-Media-Streamer (AIST)
- PSPacer (AIST) - used for stream rate limitation
- UltraGrid-Media-Player (PSNC)
- Demo-Proxy (PSNC)
- Demo-GUI (PSNC)
- Ryu OpenFlow controller (PSNC)

Demo-Proxy and Demo-GUI were developed from scratch by PSNC.

7.4.2 Step by Step Setup

The following steps are useful for high quality media streaming use case implementation:

- **Media streamer**
  - Prepare at least four different videos, each encoded using H.264 codec (we recommend at least 20Mbps bit rate for test high quality media transmissions)
  - Each video files should be included in the separate directory with proper name: directory_name+{bitrate}+Mbps (e.g. "poznan_movie_20Mbps")
  - Install UltraGrid Software v.1.3 (http://www.ultragrid.cz/) - it is recommended to use physical not virtual machine with 1G NIC
  - Connect media streamer machine to the RateLimiter
- **Rate limiter**
  - Install RateLimiter (https://code.google.com/p/pspacer/)
  - Connect RateLimiter to the local OpenFlow switch in the SDN domain
- **Madia player**
  - Install UltraGrid Software v.1.3 (http://www.ultragrid.cz/), we used media workstation with Intel i7, 12GB RAM, AMD Fire Pro 3D Graphics v8800 with four video output
  - Connect media player to the local OpenFlow switch in the SDN domain
- **Demo proxy and GUI**
  - Install demo_proxy.py from (https://github.com/ict-felix/high-quality-media-uc)
  - Open demo_proxy.py and edit the following parameters:
    - STREAM_BAND - bit rate of the video prepared for transmission

        - player_IP - list of IP address of the media player (may be the same for every stream, default up to four streams are transmitted)
        - DPID_lists - list of OpenFlow DPIDs on the path from media streamer to the media player
        - uv_streamer - paths to the UltraGrid software and UDP ports for transmissions
        - uv_player - paths to the UltraGrid software and UDP ports for transmissions, also IP address of the streamer for ping measurements, change the following switch: *-r alsa:front:CARD=Intel,DEV=0* to play audio of the media (if needed)
        - intopix_param - not mandatory, we used this parameter to control 4k projector in PSNC
        - ryu_controller_config - configuration for the Ryu OpenFlow controller together with interconnections map
        - rate_limiter_config - configuration of the rate_limiter
- **Run demo proxy**:
    - *python demo_proxy/demo_proxy.py*
    - demo-proxy automatically establish communication with media player, media streamer, rate limiter and OpenFlow controller
- **Run GUI for experiment**
    - Open web browser and put: http://{demo_proxy_ip_address}:5000
    - Perform the experiment using user interface
        - wait until OpenFlow switches register to the Ryu OpenFlow controller
        - run video stream transmissions and collect video and network statistics

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.