A developer guide to FELIX SDN experimental facilities

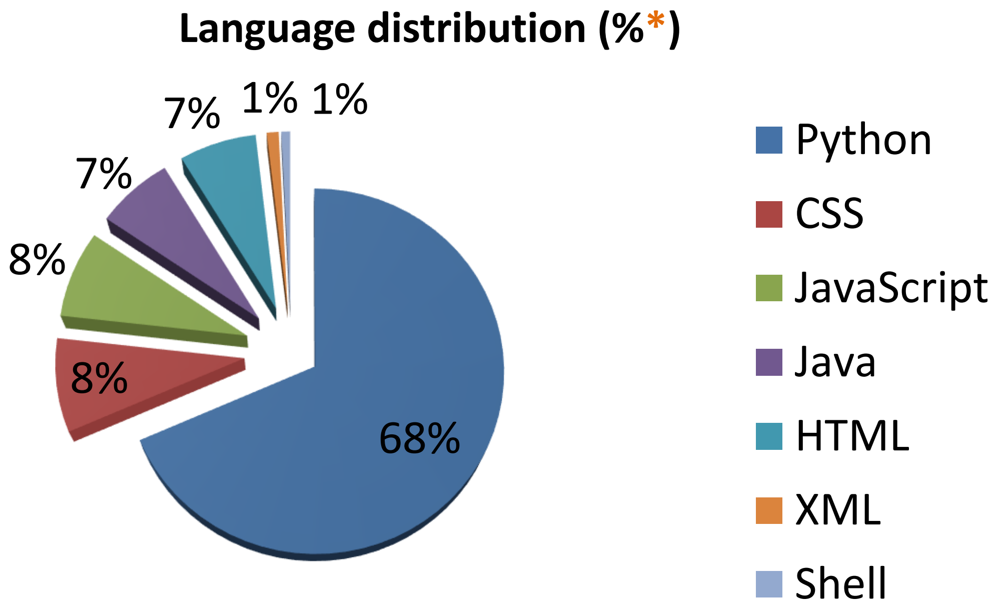
Index

FELIX as a developer: overview

- Where to find the sources:
  - https://github.com/dana-i2cat/felix
- Developer teams:
  - 1 component – 1 team
    - Integration teams may supervise components from others
  - More information:
    - https://github.com/dana-i2cat/felix/wiki/Contributing_overview#contact
- Organisation of the code:
  - 1 component – 1 branch
    - Exceptions: shared/extended software modules (e.g. CRM, C-BAS)
  - Master branch: merge of existing branches into one
    - Automated task triggered via Jenkins (ideal*)
    - Manual merge (solving conflicts)

* *Requirement: similar folder structure across branches, keeping an eye to common files at the root*

FELIX as a developer: languages and tools



- Quality check
  - 1 component – 1 syntax analysis task
  - Tasks are manually added along with new components/branches
    - Automatically triggered by pushes on branches
  - Developers can follow up the violations of code standards, LOC, complexity % or duplications in their branch

** Percentage obtained by counting no. of files per language*

FELIX as a developer: contributing (I/II)

Contributing to the code or technical documentation?

- How to proceed*:
  a) Small-sized contributions:
     1. Clone the FELIX repository in your local environment
     2. Edit the files and test behaviour. When ready, create patch(es) and send
  b) Medium to large-sized contributions:
     a) Working within GitHub
        1. Fork the FELIX repository into a new one under your account(s)
        2. Perform the improvements or fixes you'd like to contribute
        3. Start pull request against the forked repository and appropriate branch
           1. Be descriptive in your message!
     b) Working outside GitHub
        1. Clone the FELIX repository in "bare" mode (should be public)
        2. Clone the bare repository into another for modifications
        3. Edit files. When ready, commit and push to the local bare repository
        4. From modified repository, issue pull request and an e-mail contents

* *Detailed steps in https://github.com/dana-i2cat/felix/wiki/Contributing_procedure*

FELIX as a developer: contributing (II/II)

Contributing to the code or technical documentation?

- Requirements*:
  - Knowledge about Git, Python, shell
  - Understanding FELIX architecture
  - Local environment for management stack:
    - OS: Debian 7.8 (3.2.0-4)
      - Python 2.7, Pip, [Ryu/POX], [Apache/Django/MySQL], [Flask/Mongo], etc
    - Around 2Gb-4Gb RAM (depending on components in use)

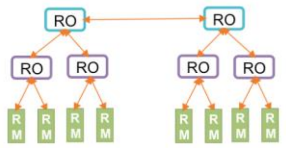
- Others:
  - Communication through issue tracker and e-mail
  - Pull requests managed by development teams
  - Name / ID / e-mail of contributors to be included in CONTRIBUTORS.md

* *Requirements in https://github.com/dana-i2cat/felix/wiki/Installation#requirements*

FELIX as a developer: spaces, architecture and modules

Architecture

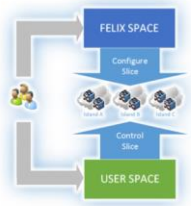
The resulting architecture attempts to be flexible and scalable enough to assure proper interaction between various system components.



The FELIX project uses a hybrid hierarchical model for inter-domain communication, where the upper layers contain a software module that aggregates information on the lower layers and act as a central point for operations such as interworking with the monitoring module.

Spaces within FELIX

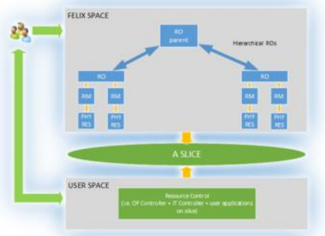
A FELIX testbed can be seen as a combination of two spaces:



- **FELIX Space**: consists of management and control tools to coordinate processes of creation of a virtual environment in a heterogeneous, multi-domain and geographically distant test-bed. The components of the FELIX space operate in a hybrid hierarchical model, to enable efficient information management and sharing across multi-domain environment.
- **User Space**: consists of any tool or application a user wants to deploy to control his virtual network environment or to execute a particular operation within it. The selection of tools is completely dependent on specific user requirements and is out of scope of the FELIX framework.

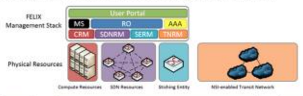
Both spaces play distinct roles in creation and operation of each virtual network environment, which is created by the FELIX Space upon a request from a user, and then managed by user tools in the USER Space.

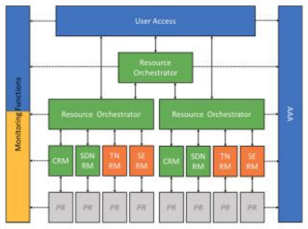
The figure below depicts a more detailed view, including the main functional blocks and their dependencies:



The FELIX Management Stack

The FELIX Space relies on the FELIX Management Stack (FMS), which consists of *Resource Orchestrators* (ROs), *Resource Managers* (RMs), the *Monitoring System* (MS) and *Authentication, Authorisation and Accountability* (AAA) modules and the physical (testbed) infrastructure to provide the resources needed for realising a user slice.





The images above show the different modules used in FELIX and their managed resources, as well as a simplified view of the relations between modules in the FELIX Management Stack.

The User Space configuration depends mostly on FELIX end user preferences and choices. The user has a freedom to choice any Resource Controller, that is suitable for his/her needs. This controller is in charge of managing the creation, modification and deletion of slices related to the experiments (including all functionality, APIs and applications) basing on resources assigned by the FELIX Space components.

Resource Orchestrator

The Resource Orchestrator (RO) is a stateful entity, responsible for the orchestration of the multi-domain service in the federated infrastructure. It coordinates the reservation and provisioning of heterogeneous network and compute resources in each domain.

The RO can work in two different modes: *normal* and *master*:

- **Normal**: typically, an RO connects to a number of RMs within the same domain. Some background tasks run periodically to detect changes in the resources provided by the underlying RMs. Also, RO provides MS with information on the physical topology that it oversees and about the slices requested by the users.
- **Master**: in an upper layer, the master RO (MRO) may oversee several ROs, e.g. at a state, continental or global level; depending on the configuration on the peers that is agreed by the members of the specific FELIX testbed. In the same manner as RO, MRO will periodically fetch resources from its peers.

Resource Managers

The Resource Managers (RMs) are software modules in charge of controlling a specific type of resource, being the equivalent of the SFA Aggregate Manager and NSA in NSI Framework.

FELIX defines four basic types of RMs:

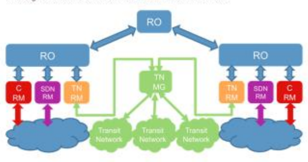
- Transit Network RM (TNRM) for the management of Transit Network
- Software-Defined Networking RM (SDNRM) for the management of SDN (here, OpenFlow)-related resources
- Stitching Entity RM (SERM) for the mapping of the two above domains
- Computing RM (CRM) for managing Virtual Machines in different hypervisors

Transit Network Resource Manager

The TNRM is focused on the on-demand multi-domain provisioning of connectivity in the transit segment. TNRM comes in two flavours:

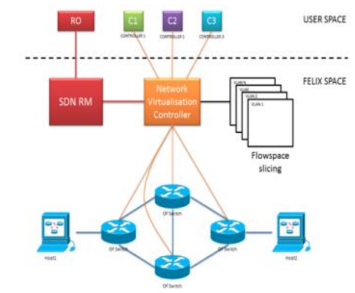
- **NSI-TNRM**
- **GRE-TNRM*

The image below shows the location of TNRM and its interactions.



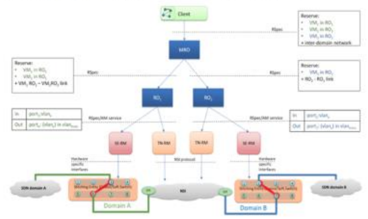
Software-Defined Resource Manager

The SDNRM controls the OpenFlow-based L2 resources. To achieve that, it uses the latest version (1.4.0-1) of the FlowVisor network controller as a proxy between the controller defined by the user in the User Space and the switches residing in the infrastructure.



Stitching Entity Resource Manager

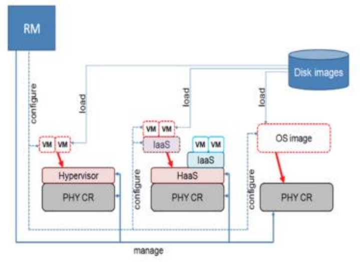
The two domains above must be related to enable communication across them. The SERM is in charge of binding the devices at the boundary at each domain.



Computing Resource Manager

The CRM module allocates (reserves) and provisions the Virtual Machines that the different islands or domains provide. It comes in two flavours, to support two different, widely used hypervisors:

- **XEN-CRM**
- **KVM-CRM**



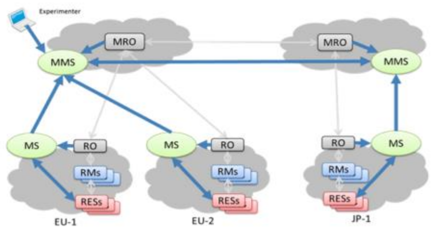
Monitoring System

The FELIX Monitoring System (MS) collects the monitoring data from the resources available per experiment or slice, including information retrieved from the different SDN islands and from the transit domains, in terms of performance of the established connectivity services.

The MS can work in two different modes: *normal* and *master*:

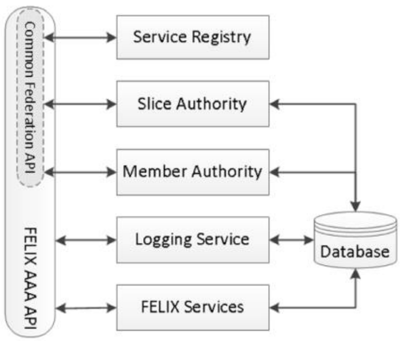
- **Normal**: the MS interacts with RO to recover information on slices and physical topology, as well as some data on the devices to monitor. MS will use such data to connect to them and fetch the expected metrics.
- **Master**: in a similar manner to RO, the Master MS (MMS) interacts with both MS and MRO to obtain aggregated measurements and other information.

The MS and MMS also interact with the Expedient user agent to provide a visual representation of the gathered metrics, orderly by time.



Certificate-based AAA for SDN Experimental Facilities

The Certificate-based AAA in FELIX is a ClearingHouse implementing the GENI Federation APIv2 and providing a registry of slices and members, among others, to exert authentication and authorisation procedures on the managed resources. For instance, CBAS provides certificates to the users and verifies whether they have or not the proper permissions or roles to request any given operation, such as listing the resources of a given RM or creating a new slice.

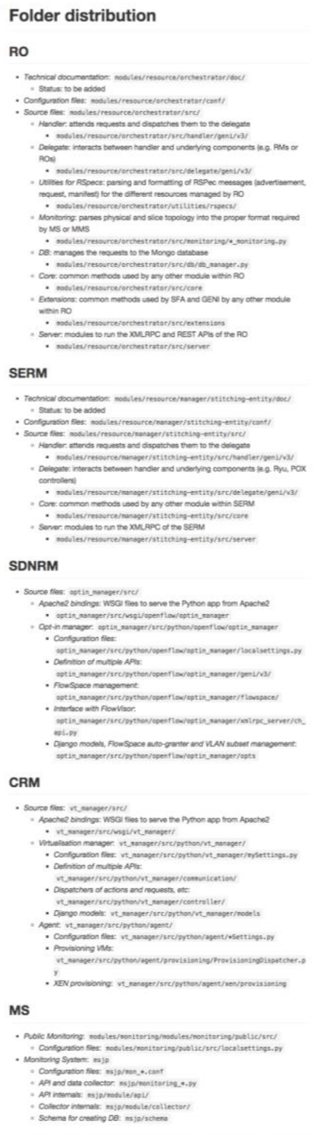


More details on architecture and its components can be found in D2.2 and D3.* in the FELIX deliverables.

* *Architecture and modules details available at https://github.com/dana-i2cat/felix/wiki/Architecture*

FELIX as a developer: distribution of the sources

- Distribution of the sources per module*:
  - **RO**:
    - Source: modules/resource/orchestrator/src/
  - **RMs**:
    - Source: modules/resource/manager/<module_name>/src/
      - *module_name: stitching-entity, transit-network*
  - **CRM, SDNRM, Expedient**:
    - Source: <module_name>/src/python/[mod_1/]<module_name>/[<mod_2>/]
      - *mod 1: -, openflow, expedient*
      - *module name: vt_manager, optin_manager, expedient*
      - *mod 2: -, -, clearinghouse*
  - **MS**:
    - Source: msjp/
  - **C-BAS**:
    - Source: src/plugins/



Folder distribution

RO

- *Technical documentation:* `modules/resource/orchestrator/doc/`
  - Status: to be added
- *Configuration files:* `modules/resource/orchestrator/conf/`
- *Source files:* `modules/resource/orchestrator/src/`
  - *Handler:* attends requests and dispatches them to the delegate
    - `modules/resource/orchestrator/src/handler/geni/v3/`
  - *Delegate:* interacts between handler and underlying components (e.g. RMs or ROs)
    - `modules/resource/orchestrator/src/delegate/geni/v3/`
  - *Utilities for RSpecs:* parsing and formatting of RSPec messages (advertisement, request, manifest) for the different resources managed by RO
    - `modules/resource/orchestrator/utilities/rspecs/`
  - *Monitoring:* parses physical and slice topology into the proper format required by MS or MMS
    - `modules/resource/orchestrator/src/monitoring/*_monitoring.py`
  - *DB:* manages the requests to the Mongo database
    - `modules/resource/orchestrator/src/db/db_manager.py`
  - *Core:* common methods used by any other module within RO
    - `modules/resource/orchestrator/src/core`
  - *Extensions:* common methods used by SFA and GENI by any other module within RO
    - `modules/resource/orchestrator/src/extensions`
  - *Server:* modules to run the XMLRPC and REST APIs of the RO
    - `modules/resource/orchestrator/src/server`

SERM

- *Technical documentation:* `modules/resource/manager/stitching-entity/doc/`
  - Status: to be added
- *Configuration files:* `modules/resource/manager/stitching-entity/conf/`
- *Source files:* `modules/resource/manager/stitching-entity/src/`
  - *Handler:* attends requests and dispatches them to the delegate
    - `modules/resource/manager/stitching-entity/src/handler/geni/v3/`
  - *Delegate:* interacts between handler and underlying components (e.g. Ryu, POX controllers)
    - `modules/resource/manager/stitching-entity/src/delegate/geni/v3/`
  - *Core:* common methods used by any other module within SERM
    - `modules/resource/manager/stitching-entity/src/core`
  - *Server:* modules to run the XMLRPC of the SERM
    - `modules/resource/manager/stitching-entity/src/server`

SDNRM

- *Source files:* `optin_manager/src/`
  - *Apache2 bindings:* WSGI files to serve the Python app from Apache2
    - `optin_manager/src/wsgi/openflow/optin_manager`
  - *Opt-in manager:* `optin_manager/src/python/openflow/optin_manager`
    - *Configuration files:* `optin_manager/src/python/openflow/optin_manager/localsettings.py`
    - *Definition of multiple APIs:* `optin_manager/src/python/openflow/optin_manager/geni/v3/`
    - *FlowSpace management:* `optin_manager/src/python/openflow/optin_manager/flowspace/`
    - *Interface with FlowVisor:* `optin_manager/src/python/openflow/optin_manager/xmlrpc_server/ch_api.py`
    - *Django models, FlowSpace auto-granter and VLAN subset management:* `optin_manager/src/python/openflow/optin_manager/opts`

CRM

- *Source files:* `vt_manager/src/`
  - *Apache2 bindings:* WSGI files to serve the Python app from Apache2
    - `vt_manager/src/wsgi/vt_manager/`
  - *Virtualisation manager:* `vt_manager/src/python/vt_manager/`
    - *Configuration files:* `vt_manager/src/python/vt_manager/mySettings.py`
    - *Definition of multiple APIs:* `vt_manager/src/python/vt_manager/communication/`
    - *Dispatchers of actions and requests, etc:* `vt_manager/src/python/vt_manager/controller/`
    - *Django models:* `vt_manager/src/python/vt_manager/models`
  - *Agent:* `vt_manager/src/python/agent/`
    - *Configuration files:* `vt_manager/src/python/agent/*Settings.py`
    - *Provisioning VMs:* `vt_manager/src/python/agent/provisioning/ProvisioningDispatcher.py`
    - *XEN provisioning:* `vt_manager/src/python/agent/xen/provisioning`

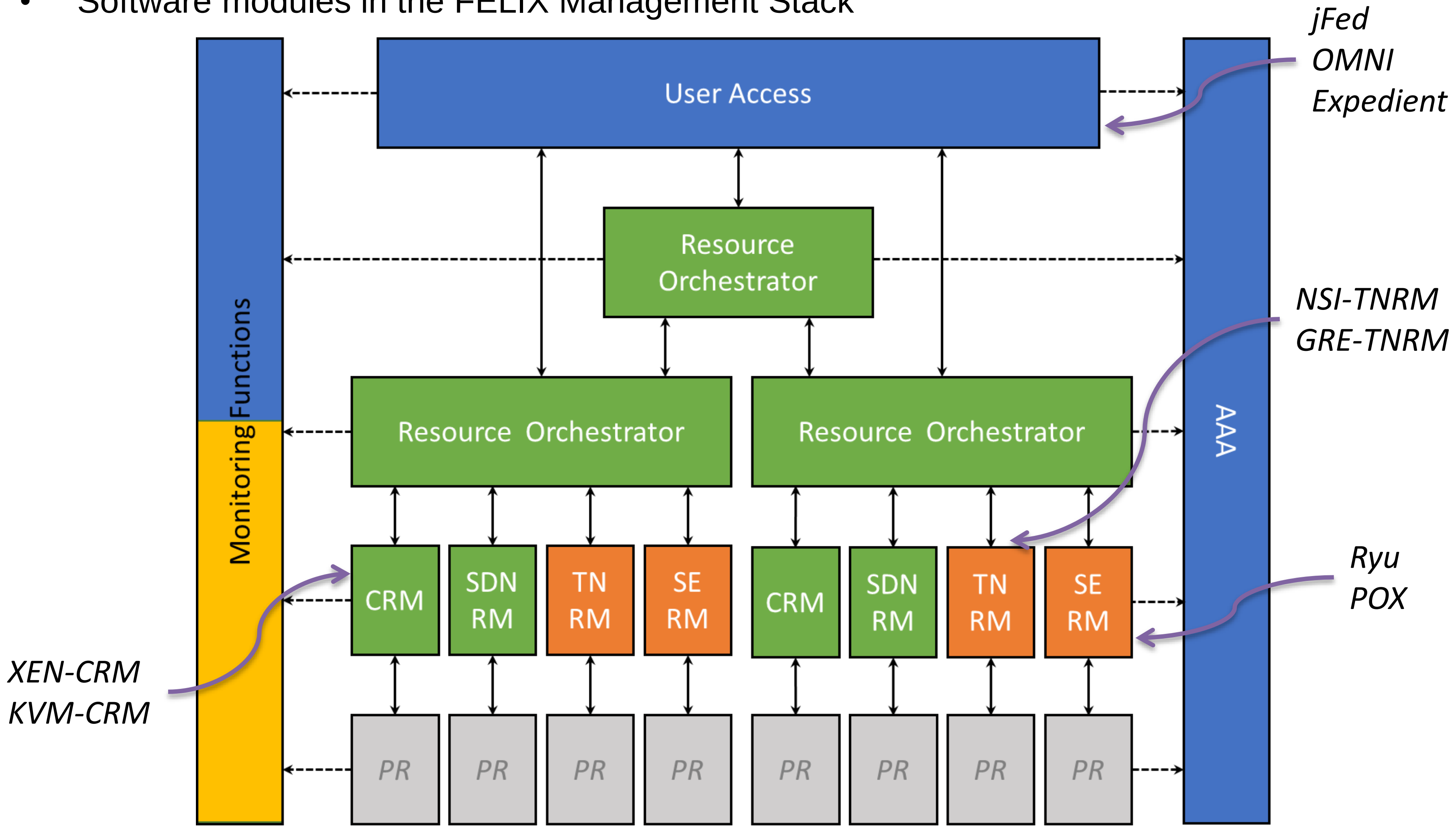
MS

- *Public Monitoring:* `modules/monitoring/modules/monitoring/public/src/`
  - *Configuration files:* `modules/monitoring/public/src/localsettings.py`
- *Monitoring System:* `msjp`
  - *Configuration files:* `msjp/mon_*.conf`
  - *API and data collector:* `msjp/monitoring_*.py`
  - *API internals:* `msjp/module/api/`
  - *Collector internals:* `msjp/module/collector/`
  - *Schema for creating DB:* `msjp/schema`

* *Details of source location available at https://github.com/dana-i2cat/felix/wiki/Contributing_development_info#folder-distribution*

FELIX infrastructure: overview of software

- Software modules in the FELIX Management Stack



FELIX infrastructure: preparing the environment

- Clone the following repositories under `/opt/felix`:
  - *Note: each component/branch under folder with its name**
  - https://github.com/dana-i2cat/felix.git
    - Repeat per branch/component
  - https://github.com/EICT/C-BAS.git

- Requirements:
  - OS: Debian 7.8 (3.2.0-4)
  - Around 2Gb-4Gb RAM (varies on number of machines used to host the modules)
  - Packages: Python 2.7, Pip
    - **RO**: Flask, MongoDB, Flup, lxml, apscheduler, ...
    - **SERM**: Flask, MongoDB, Ryu/POX
    - **CRM, SDNRM, Expedient**: Apache, Django, MySQL, FlowVisor
    - **TNRM**: Apache CXF, Jython, NSIv2
    - **MS**: Apache2, SQLAlchemy, Bottle, Elixir, gevent, ...
    - **C-BAS**: Flask, MongoDB, SQLAlchemy, flup, swig, xmlsec1, ...

* *Each component/branch shall be downloaded individually (clone + checkout)*

FELIX infrastructure: setting up the facility (I/II)

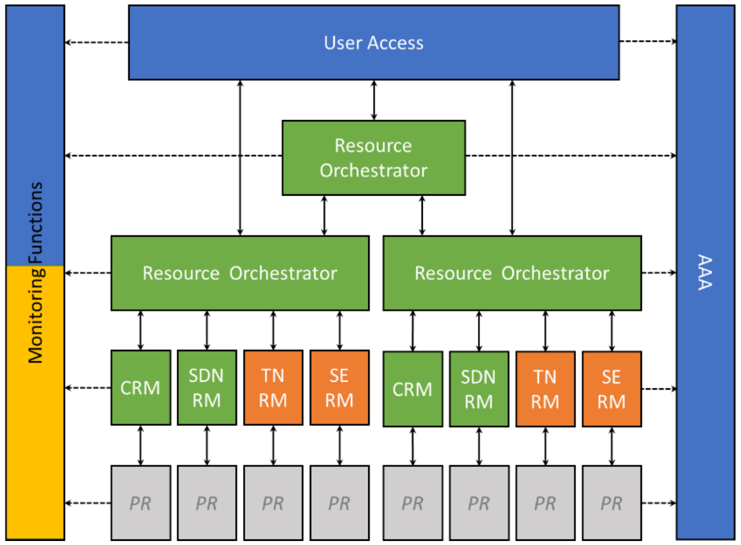
- Installation* steps:
  1. **RMs**

     Lower layer (management)
     1. SDNRM
     2. XEN-CRM / KVM-CRM
     3. SERM
     4. NSI-TNRM / GRE-TNRM
  2. **MS/MMS**

     Transversal layer (interacts with physical resources)
  3. **CBAS/MCBAS**

     Transversal layer (trust anchor)
  4. **RO/MRO**

     Medium and upper layers (logic management)



** Installation instructions available at https://github.com/dana-i2cat/felix/wiki/Installation and per module*

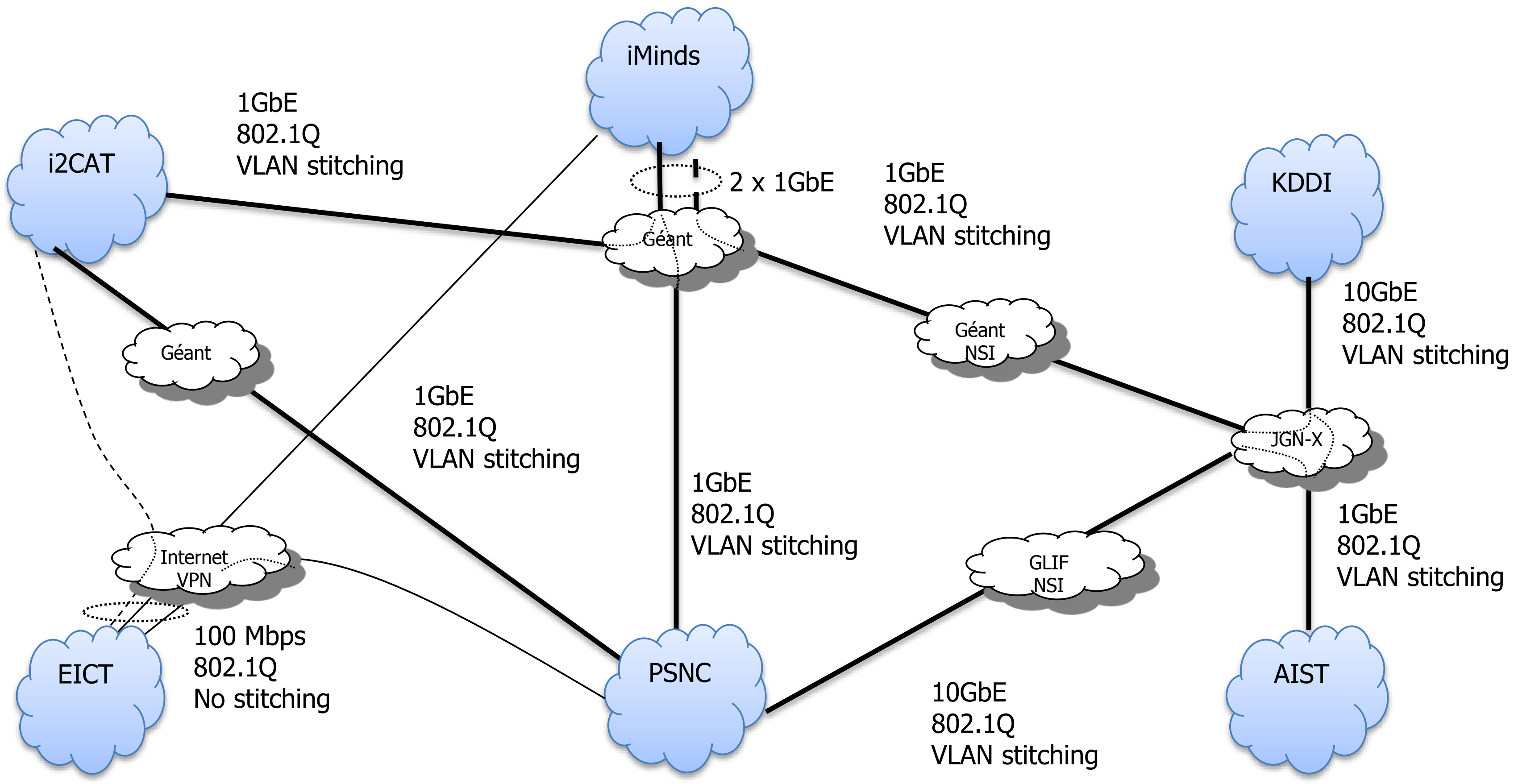
FELIX infrastructure: setting up the facility (II/II)

- Configuration* steps:
  - **RM**
    - Set up SDN environment: set up FlowVisor and connection with SDNRM
    - Configure virtualisation servers: set up hypervisor and connection with CRM
    - Define SERM endpoints and links with SDN and TN domain
    - Configure TNRM with NSI connection and endpoints or GRE tunnel
  - **MS/MMS**
    - Identify devices to be monitored by MMS
    - Point to MMS (or appoint as MMS*)
  - **CBAS/MCBAS**
    - Configure with name of island/domain, issue certificates, establish trust between partners
    - Point to MCBAS (or appoint as MCBAS*)
  - **RO/MRO**
    - Configure with details on some managed resources, parameters, etc
    - Point to MS (if RO) or to MMS (if MRO)
    - Appoint as MRO*

**Configuration instructions available per module* **Reconfiguration upon agreement*

FELIX infrastructure: overview of networking

- Network connectivity among islands in the FELIX federation



FELIX infrastructure: federating with other islands

- Federating implies (at least):
  - **Certificate** exchange
    - Trusting others' RMs
    - Trusting others' CBAS (and its managed entities)
    - Being trusted by MCBAS
  - **Monitoring** information exchange
    - Exposing topology + slice information from island's MS
    - Sending metrics and above information to MRO and MMS
  - **Networking**
    - Being connected to the other islands:
      - VLANs, NSI domain, etc.
  - Continued support to ensure operations on owns' island:
    - Solve issues on either networking or software domains

➢ Agreement among existing sites with new one

➢ Manual configuration and set-up required for federation

- Rearrangement/*relocation* of MRO, MCBAS, MMS implies small config. changes

FELIX infrastructure: support to infrastructure set-up

- Support*
  - Installation and configuration
    - Provided by specific developer team
  - Federation (networking, certificates, etc.)
    - Provided by specific infrastructure team

** Contact details available at https://github.com/dana-i2cat/felix/wiki/Contributing_overview#contact*

PARTNERS

Poznan Supercomputing
and Networking Center
Poland

National Institute
of Advanced Industrial Science
and Technology
Japan

Nextworks
Italy

Fundacio Privada i2CAT,
Internet I Innovacio Digital
A Catalunya
Spain

SURFnet bv
Netherlands

European Center for Information and
Communication Technologies Gmbh
Germany

iMinds VZW
Belgium

KDDI
Japan