The SUNRISE Gate

1. Architecture

The SUNRISE GATE is the interface through which users can run their experiments, accessing in a unified way the heterogeneous resources offered by the testbeds of the SUNRISE federation. In the reference architecture shown in Figure 1, testbeds are accessible via gateways (GWs). The interaction between the GWs and the GATE is managed by suitable plug-ins capable to communicate with the GWs and to handle the specific characteristics of the testbeds. Plug-ins are also in charge of conveying the data gathered by the GW in order for the GATE to offer the user a unified and standard view. Notice that capabilities and data are accessed by users employing a unified interface irrespectively of the complexity and specific characteristics of the underlying testbeds. Similarly, experiments are defined using a standard interface.

In particular, the SUNRISE GATE has been designed to provide Web interfaces both in the form of Web GUIs, designed to be effective for users with a limited experience in ICT technologies (e.g., a researcher in marine biology), and Web Services that allow users with ICT skills to develop more advanced and customised solutions. However, in some cases the Web interfaces are not well suited to access all the low-level functionalities offered by the devices in the testbeds. As an example, updating the firmware on a modem could be significantly easier and more effective when a direct access through a suitable Operating System shell is available. In this perspective, the user is offered the possibility to access the GWs via a shell interface, by means of a preliminary authentication, authorization and accounting performed at the SUNRISE GATE.

Clearly, before using them, the resources employed in an experiment have to be scheduled. For sake of simplicity, when an experimentation period starts and consequently resources on one or more testbeds are reserved, the other simultaneous experiments cannot reserve resources on that testbed. In other words, experiments cannot be run in parallel on the same testbed. Notice that this is also necessary to avoid interference among experiments that could compromise the quality of the results.

More formally, an experimentation period is fully qualified by a tuple with the following information $<start_{exp}, end_{exp}, IP_{testbed\ 1}, ..., IP_{testbed\ n}>$, where $IP_{testbed\ i}$ is the IP of the GW of the i^{th} testbed involved in the experiment and $start_{exp}$ and end_{exp} are the instants in time in which the experiments starts and ends.

A simplified sequence of the necessary steps to deploy an experiment is the following:

1) GWs announce to the GATE their capabilities and all relevant information to configure experiments (e.g., positions of the nodes, environmental conditions, etc);

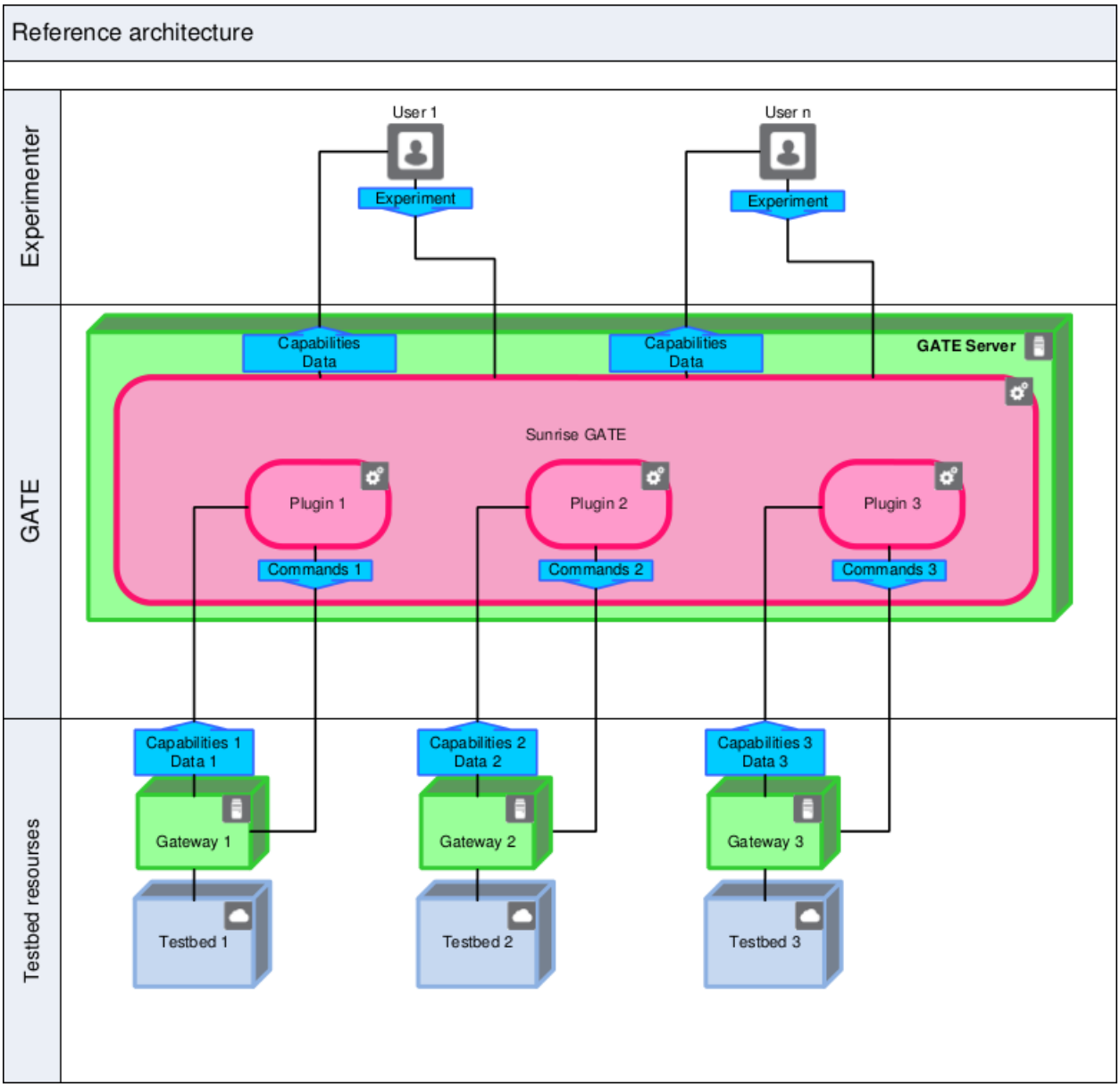


Figure 1 - Reference architecture

2) The user accesses the GATE to know the capabilities offered by the federation and the availability of the testbed resources based on a calendar.;
3) The GATE allows the user to access the resources during the period of time booked for her/him;
4) The user can then configure the resources and issue the commands needed to run experiments;
5) Data collected in the testbeds during the experiment are presented to the user;
6) When the period allocated to perform the tests ends, resources are released and can be used for the next experiments.

In Figure 2, we present the SUNRISE architecture depicted in the FED4Fire (INSERT LINK: http://www.fed4fire.eu/) notation.

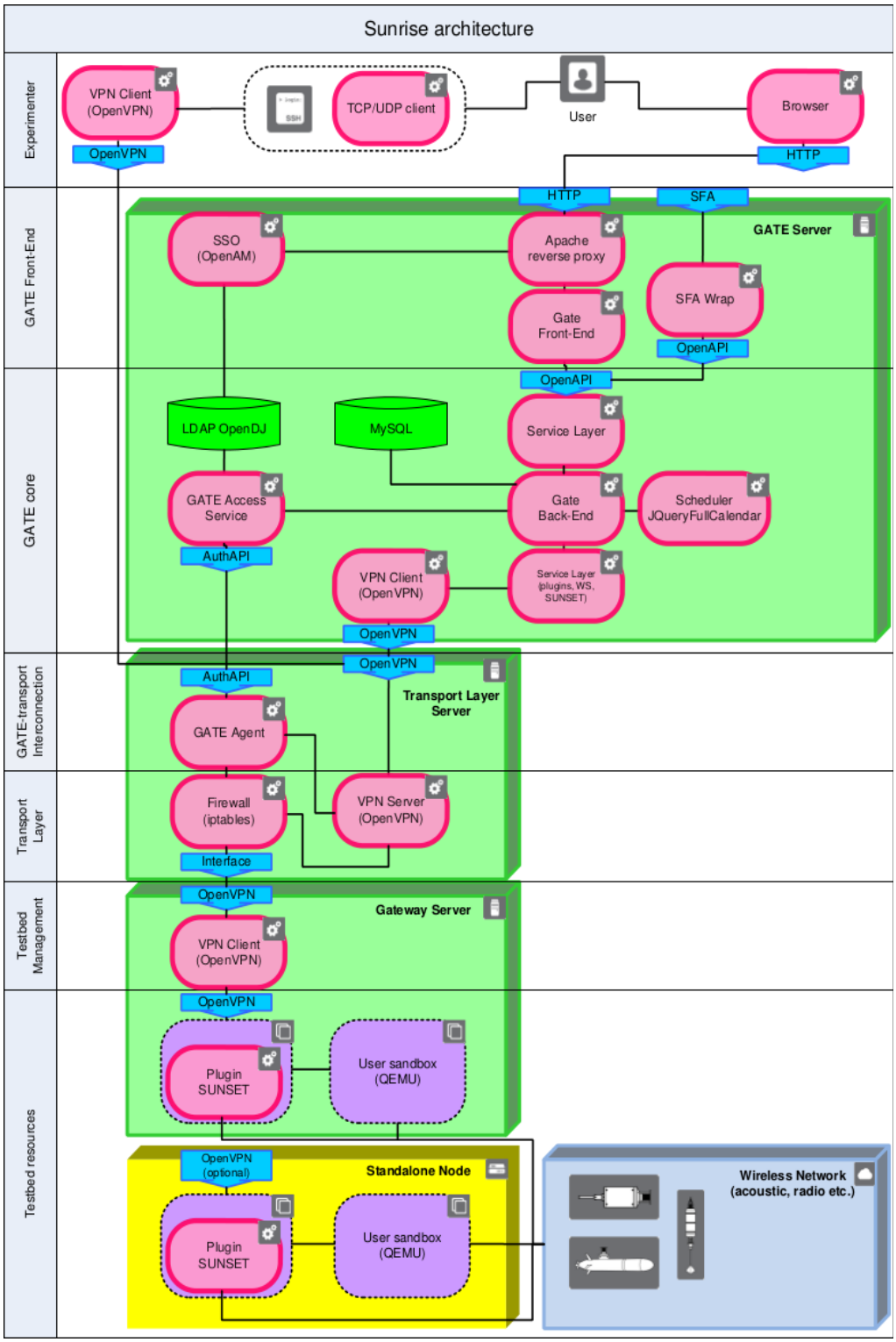


Figure 2 – Architectural elements and interfaces.

For the sake of simplicity, we focus on the Sapienza University Networking framework for underwater Simulation Emulation and real-life Testing (SUNSET) plug-in (see Section 1.4). Notice that SUNSET allows us to access most of the functionalities provided by the testbeds in the SUNRISE federation.

In Figure 3, the technologies used for the implementation of the system are presented aside the boxes representing single components the architecture.

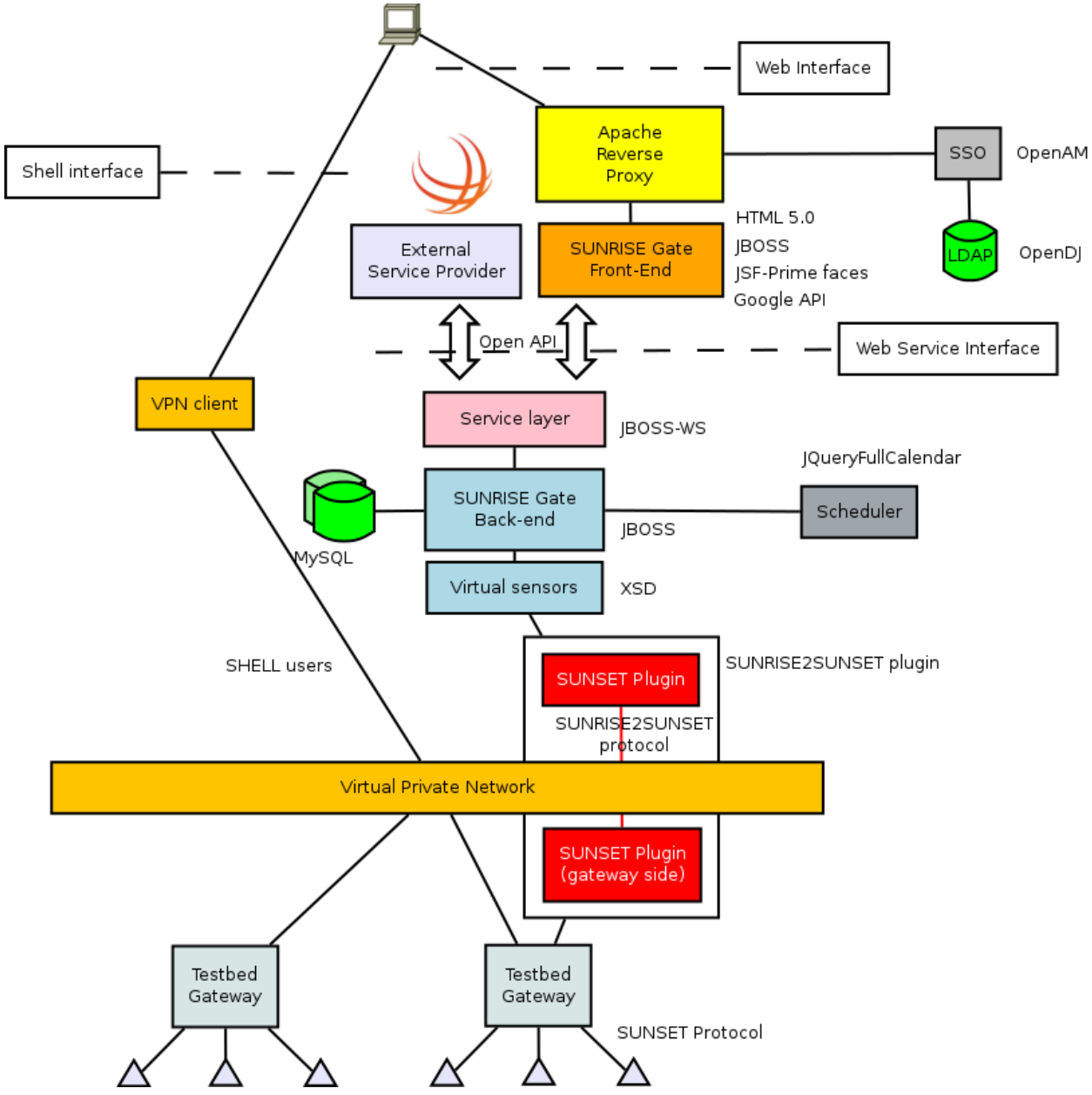


Figure 3 - Technologies used for the implementation of the system.

In the following we illustrate the SUNRISE GATE architecture, from the perspective of the interfaces to access the gate services, namely the Shell, Web GUI and Web services interfaces.

1.1 Shell Interface

The shell is accessed through a VPN to guarantee the highest level of security. When the OpenVPN client tries to access the Shell (that is configured as an OpenVPN server), the user credentials are forwarded to the Single Sign On (SSO) module of the SUNRISE GATE. The SSO module checks the credentials with the LDAP (Lightweight Directory Access Protocol) authentication and provides back an access token, or denies the access to the testbeds.

The access token is the tuple that uniquely defines the experiment, namely $<start_{exp}, end_{exp}, IP_{testbed\ 1}, ..., IP_{testbed\ n}>$. The VPN will be active only in the interval in time defined by $start_{exp}$ and end_{exp}, having as end-points the client and the IP of the Shells (i.e. the GWs) defined in the access token.

Since in this document we illustrate the integration of testbeds powered by SUNSET, the use of the shell will allow the users to access all the functionalities provided by SUNSET that will be discussed in section 1.4. However, the same architecture can be used to interact with other plug-ins.

For accessing the advanced functionalities, a VPN shell is to be open on every GW involved in the experiment, using a SUNRISE account, centrally verified on RADIUS. Web shell is an alternative to VPN shell.
For accessing Web functionalities, the same credentials as for the advanced functionalities have to be used.

1.2 Web GUI Interface

The user can access the Web GUI interface by providing credentials to the SUNRISE GATE. Apache Reverse Proxy is used to hide from the user the underlying structure and complexity; it retrieves the resources on behalf of the Web client from the servers in the federation and returns the resources as though they originated from the proxy itself.

The SUNRISE GATE Front-End employs modern technologies, such as HTML 5.0, JSF Prime Faces and Google API, to implement an effective Web GUI specifically designed to simplify the access to the federation for non-ICT expert users. Figure 4 shows a sequence diagram of the user interaction for issuing an experiment.

Clicking on a node displayed on a map of the testbed, information on the location, the hardware and software of the node as well as the topology and link quality are visualized in a pop-up window.

The information visualised on the front-end, and more in general the interaction with the GW and consequently with the testbeds, is abstracted in a Service Layer that implements the Web Service Interface described in the next section.

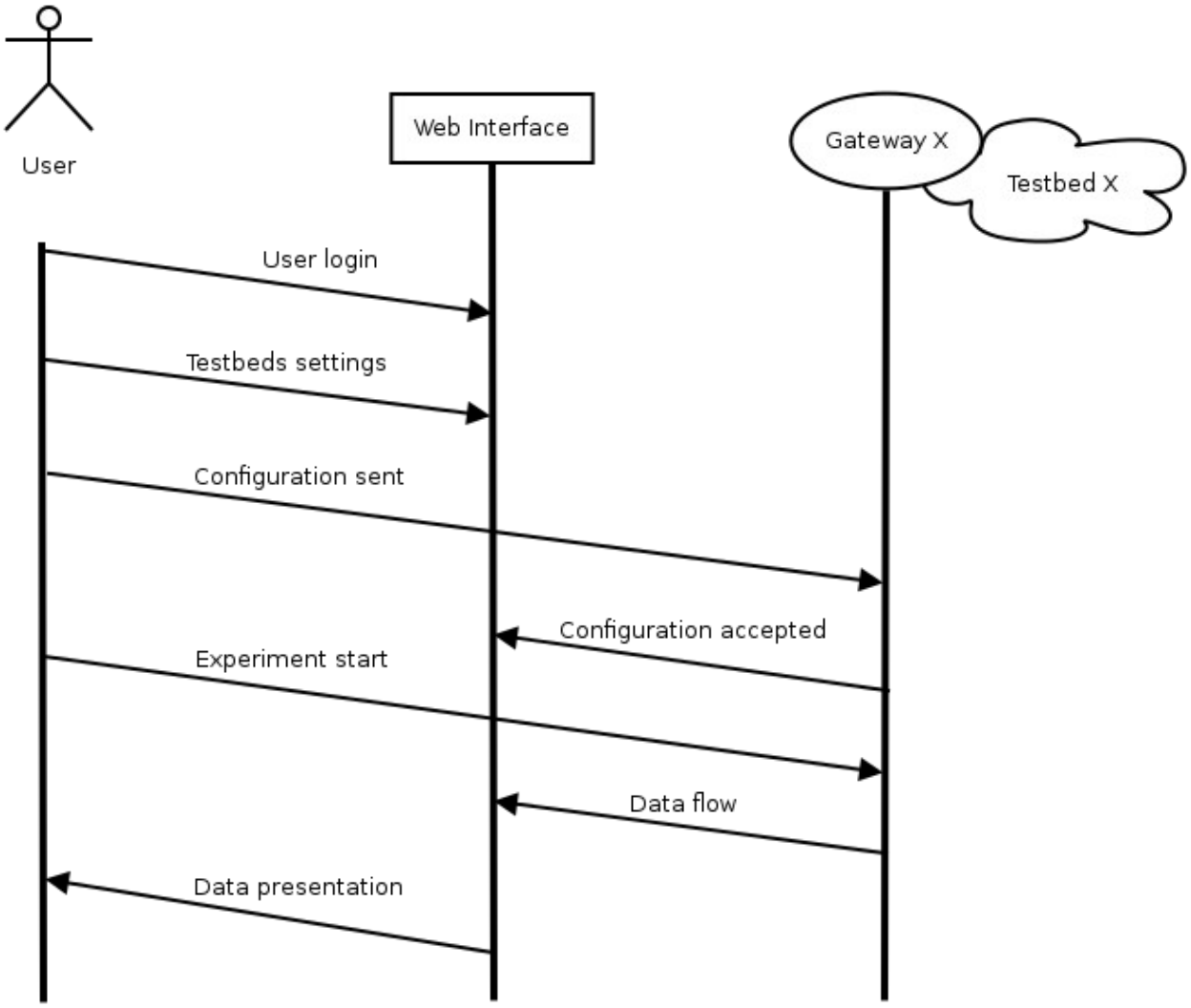


Figure 4 - Sequence diagram of the user interaction for issuing an experiment.

1.3 Web Services Interface

The Service layer offers open APIs through which both external service providers, such as FED4fire (LINK TO: http://fed4fire.eu/), and the SUNRISE gate front-end, can access the services offered by SUNRISE federation (see Figure 5). The Service layer and the SUNRISE GATE back-end run on the open source application server JBOSS capable of supporting JAVA EE on multiple platforms. The back-end provides a persistence layer implemented on MySQL to store relevant data and information on the testbeds and on the experiments.

Furthermore, the back-end manages the scheduler of the experiments, namely the component that schedules, reserves and releases the necessary resources for the whole duration of the experiment.

Finally the back-end manages a number of plug-ins that are responsible for the interaction with specific GWs (i.e. testbeds). In the next section we discuss in some more details the SUNRISE2SUNSET plug-in.

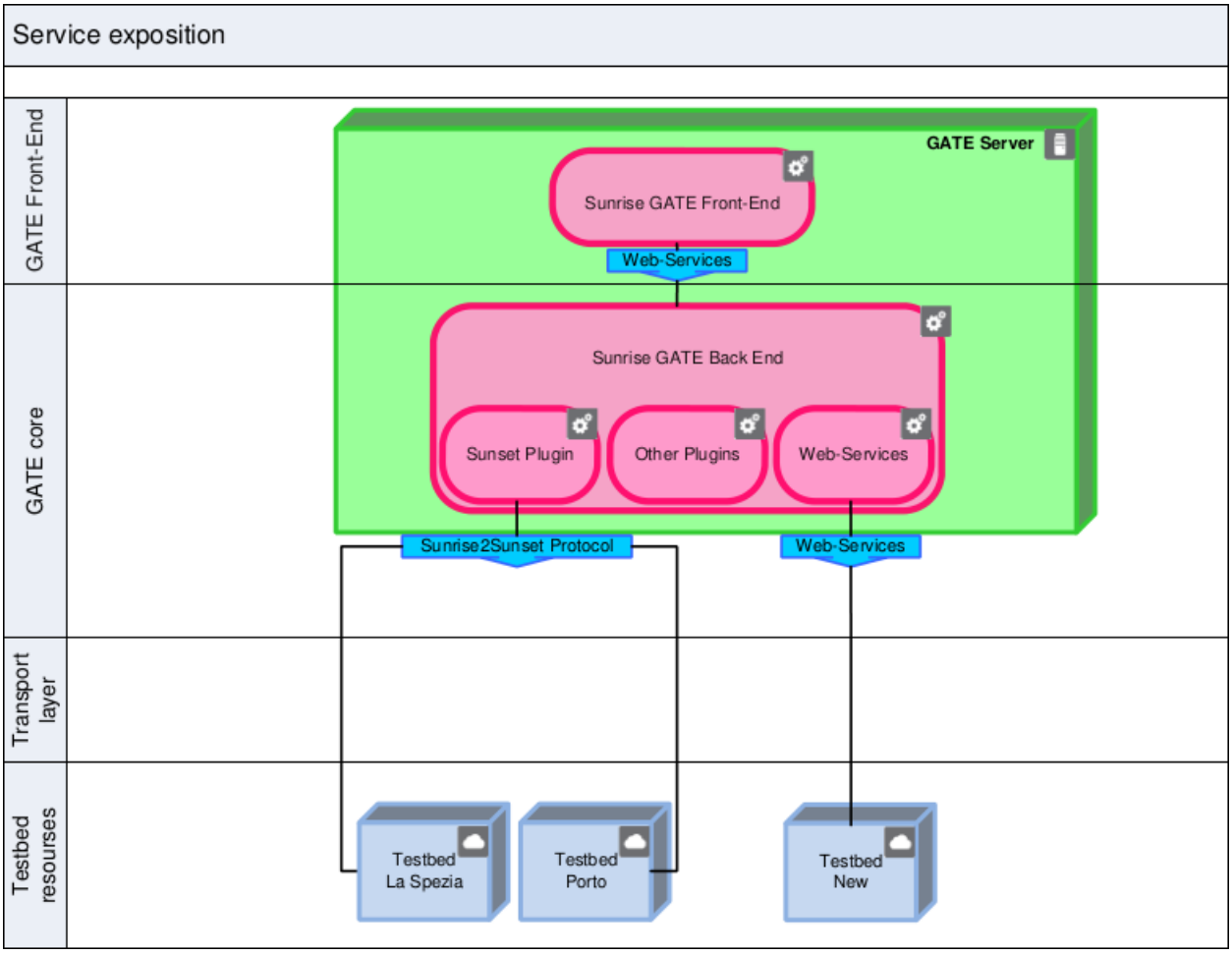


Figure 5 - Service exposition.

1.4 SUNRISE2SUNSET Plug-in

Each SUNRISE plug-in is made of the protocols and interfaces that are used to interact with the SUNRISE daemons running on the GWs.

The resources available in the testbed are described in the Virtual sensor component, namely an XML file that describes all the features, attributes, data, properties and functionalities of the resources in the testbed. Heterogeneity is hence made transparent to the upper layers, as every resource is treated as a Virtual sensor, including simulated resources. Clearly, the XML Schema can evolve with time if new resource types are introduced.

Each plug-in is in charge of abstracting all the underlying heterogeneity and complexity, providing to the upper layers a unified, standard and abstract view. Multiple plug-ins can be supported and each of them can provide and support different functionalities and capabilities.

SUNSET has been selected as the standard plug-in for the SUNRISE project. The SUNRISE2SUNSET plug-in includes the GATE-side SUNSET plug-in and the GW-side SUNSET plug-in, communicating via a protocol, namely SUNRISE2SUNSET protocol.

SUNSET has been the first framework based on open source software to seamlessly simulate, emulate and actually test (at-sea) novel underwater systems. It allows researchers and developers to easily design, implement, validate and evaluate novel communication protocols for underwater sensor networks with the support of a complete tool chain from simulation to in field tests. SUNSET has been largely enhanced, improved and validated in the past four years during more than fifteen in field campaigns.

The main task of the SUNSET plug-in is to instruct and interact with the heterogeneous set of nodes in the testbed (cabled, at the surface, underwater, etc.) according to the commands and requests received at the GW.

SUNSET translates (compressing and adapting) all the messages coming from the GW in a format that can be actually transmitted in water and used by the various underwater assets and supports the networking capabilities to reach all the nodes in the network in a distributed and efficient way.

Multiple communication interfaces (radio and acoustic) are supported for the interaction with nodes at the surface or underwater.

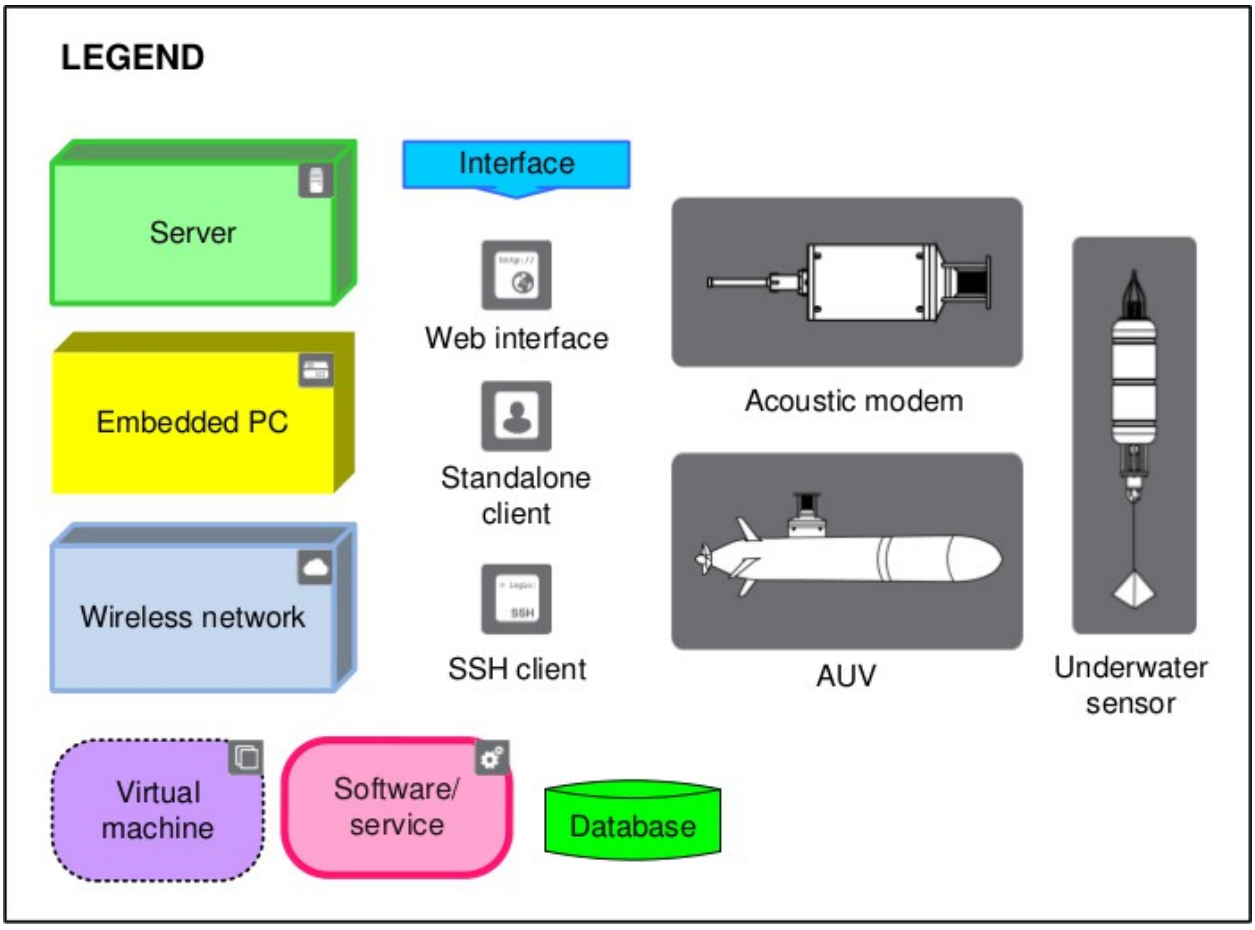
Thanks to SUNSET, the GW, and therefore the GATE, have a continuous control on all the deployed assets in real-time. Among the functionalities and capabilities provided by SUNSET we have: Discovery of all the nodes and resources available in the testbed; collection of topology information such as link qualities, ranges, neighbour lists, etc.; reconfiguration of the control software of underwater assets; data collection and reconfiguration of the underwater sensors; control of underwater mobile robots; set up, running and monitoring of various networking experiments involving multiple protocol solutions at the different layers of the protocol stack; etc.

Additionally, the SUNSET plug-in provides the support for both the interaction via Web GUI Interface, for standard users, and via a control shell, for advanced users, with more complex and enhanced commands and functionalities.

In SUNRISE, SUNSET has been further improved and enhanced to support all the features and capabilities required by the project thus providing a networking framework that is much more flexible and complete than its original version. This process will continue for the entire duration of the project.

Moreover, SUNSET currently implements the first prototype of the SUNRISE Software Defined Communication Stack (SDCS), enabling the possibility to adapt, at run time, the protocol stack configuration and tuning, making use of multiple solutions at each layer of the protocol stack. This selection and tuning can be done according to the dynamics of the underwater acoustic channel, to the resources of the nodes and of its neighbours, etc. to improve the overall networking performance.

Decision policies on how to adapt the protocol stack according to these phenomena are currently under investigation as part of SUNRISE activities and are being integrated in SUNSET. Additionally, SUNSET will support the possibility to run at the same time open source solutions and legacy and proprietary closed solutions. This approach will allow SUNSET to be open to the industrial community that may not be interested in sharing its research and solutions with the rest of the scientific community.



2. Use cases

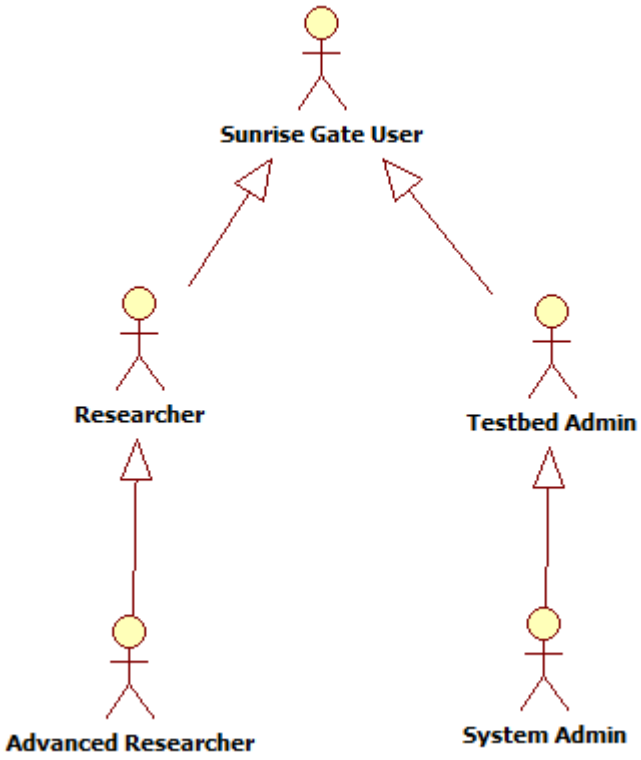


Figura 6: Actors

UC01: Make Testbed Discovery.

To be part of the SUNRISE system, a testbed must be registered with the SUNRISE server. The registration of a new testbed is divided into two phases: the first phase is the “first contact” phase when the gateway presents itself to the server providing the required contact information, such as its IP address, the listening port and the communication protocol used; once the server has successfully registered the gateway, the second phase begins, namely the “discovery” phase, through which the server instructs the gateway to start the discovery.
With this procedure, information about testbed structure and commands that can be executed on nodes is retrieved.

The registration can be done in two ways:

- Manual: through the web interface a testbed administrator can insert the required information (first phase) to reach the intended gateway. Then the discovery will be instructed to the intended gateway immediately by the server.

- Automatic: the gateway contacts the SUNRISE server exchanging the required contact information. Only after receiving the discovery request, the gateway can provide the discovery information to the server.

Automatically and at regular interval or depending on certain conditions, the system executes the discovery again.

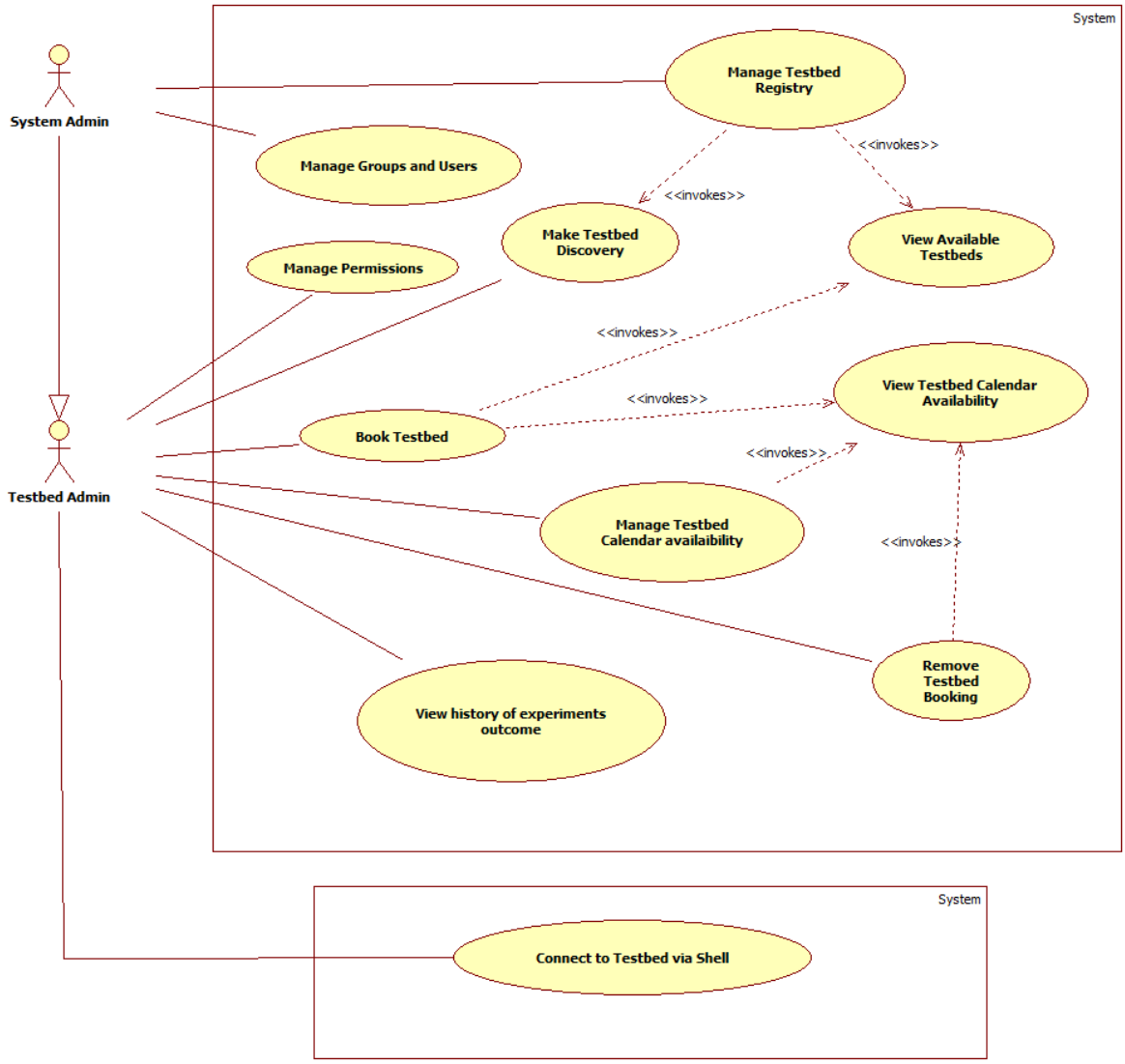


Figura 7: Admin Use Cases

UC02: Manage Testbed Registry

A system administrator manages the list of available testbeds for the experiments. Testbeds are included into the system through a Discovery procedure. A testbed is composed of N nodes. For

every node, sensing or actuating capabilities can be available. For each of the available capabilities, it is possible to execute commands.
It is possible to inspect the structure of a testbed down to the level of executable commands for each single capability.

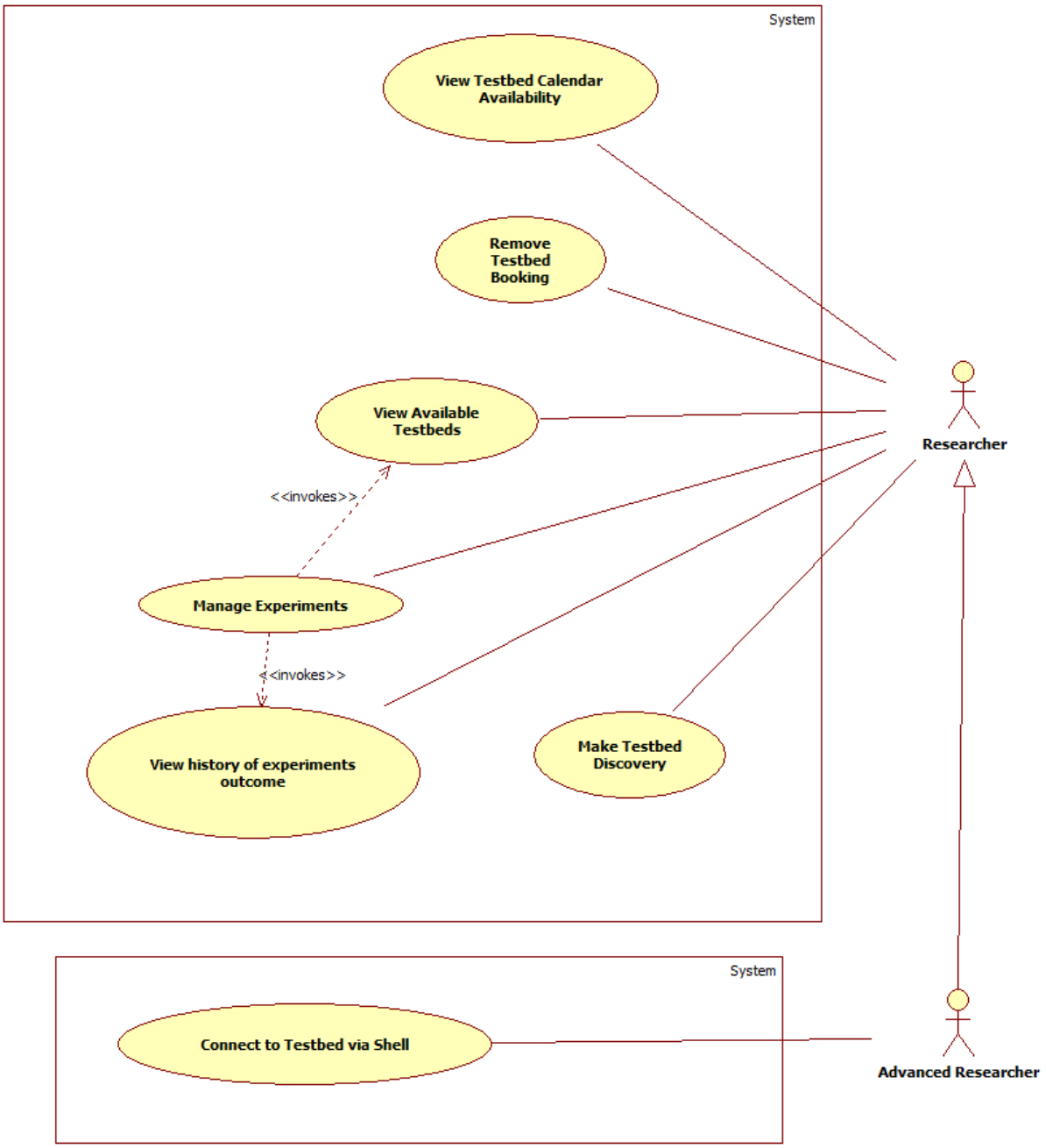


Figura 8: Researcher Use Cases

UC03: Manage Testbed Calendar Availability

The administrator is responsible for setting the available dates on the calendar for every single testbed. Available periods of a testbed will be the periods of availability defined by the administrator minus the already booked dates.
The administrator can see the list of all bookings, both active and spent bookings.
If the administrator disables/removes a user from SUNRISE, then the system cancels (logical cancellation) all the bookings associated with her/him.

UC04: Book Testbed

Each testbed has its own “booking period” as decided by its owner. A user can see the availability period of a testbed and can book it inside its booking period. Together with the booking time, a user must provide some other information such as the assets that he or she wants and some general information about the intended use.
Once booked, the testbed will be exclusively and entirely assigned the user that has booked it. Two concurrent bookings for the same testbed are not possible.

UC05: Remove Testbed Booking

An administrator can cancel the booking in two ways:

- "directly", by using a specific functionality available on the web application for managing bookings.
- "indirectly", as a consequence of having put a testbed in the "DISABLED" state.

An administrator of a testbed can cancel a previously made booking.

UC06: View history of experiments outcome

A researcher and an administrator of a Testbed can access the experiments already made by all the members of the same group.

UC07: Manage Experiments

High-level command plan: Sequence of commands the user can execute on the capabilities of the node.
Experiment: The set of the command plans the user submits throughout the period of use of the testbed.

The outcomes of high-level commands automatically get from the testbed.
When the time for a testbed reservation comes, the user who booked the testbed is granted full access to testbed resources and she can hence issue commands to be sent to the nodes.
The experiments, i.e. the settings, can be configured and executed in two ways:

- Web interface;

- Remote Shell.

With the web interface the user can access a graphical interface and can formulate and send commands to the nodes in an easy and user-friendly way. By clicking on the nodes in the map, the user accesses the capabilities of that node and fills in all requested parameters, through an easy walk-through.
The use of the remote shell is intended for researchers that are familiar with the programming languages used in the testbed. The user has got direct contact to the nodes and can issue commands to the nodes with the complete set of parameters.

A Web Experiment is performed via a web form, based on the node information that is reported in an XML file. The XML file represents the node in the SUNRISE GATE and follows a specific XSD Schema.

New settings, overriding the current one, can be sent at any time, during the time scheduled for the experiment.
An experiment is executed on all the involved testbeds. Command plans are logged.
Collected data can be visualized in near real time (NRT).
During a Shell Experiment, a researcher configures and run commands via SSH shell, directly on the testbeds.

3. Data model

The result of the discovery is a SetType composed by an id attribute, a name and a type that describes the type of the discovered network. Only the name is a mandatory attribute coming from the discovery. All the ids are mandatory but internally computed. The SetType has an element of type NetworksType.

The NetworksType is composed by an id attribute and a list element of NetworkType.

The NetworkType represents the access point for the discovered network. It has some attribute: an id, the listening port, the IP address, the protocol used for the communication (plug-in or Web Service), its name and its type. It has a PositionType element and a NodesType element. All attributes and the position are mandatory and come from the initial message of the discovery.

The PositionType has an id and the three attributes that characterize it. The NodesType has the id and a list element of NodeType representing each node of the network.

Each NodeType is composed by an id, coming from the discovery and mandatory, a GeneralInfoType describing the general information of the node, a TopologyInfoType representing the relative view of the network with respect to this node, a

HardwareInfoType describing the hardware mounted on this node, a SoftwareInfoType for the software part and the PlainInfoType.

In the GeneralInfoType there are the name of the node, its position and type. The type of the node can be a StaticType and a VehicleType. For a static node there is the definition of its typology. For each vehicle, typology and vendor are specified and a list of commands inside the CommandsType are provided.

Each command has its name as an attribute and a list of input parameters, if needed, and output response, if available.

The TopologyInfoType contains the information of the relative view of the network. NeighborsType contains the list of neighbours, and for each of them the link qualities and range information.

Each link quality element has a source, a destination and the value element. The same applies to range information where the value element represents the distance between the source and the destination.

The HardwareInfoType is composed by a list of HardwareCategoryType.

Each of them is composed by an attribute name describing the category and a list of HardwareType describing the real hardware mounted on the node.

Each HardwareType is defined by a model, vendor and version and being a complex hardware is composed by more ComponentTypes, representing each simple hardware component.

Also the ComponentType is characterized by a model, version, vendor and type attribute. It has an element MeasuresType containing a list of the possible measure that this component can do and a SettingsType containing the possible settings.

A MeasureType is composed by an id, the name of the measure, the type of data retrieved and the unit of measure.

A SettingType is composed by an id, the name, a possible minimum and maximum value of setting, a default value, the discovered value, the visible and accessible attribute for web interface composition and the type attribute of the setting value.

The SoftwareInfoType describes the MAC protocols available in the node and listed in MAC_availableType and the routing protocols available in the node and listed in Network_availableType.

4. Entity relationship diagram

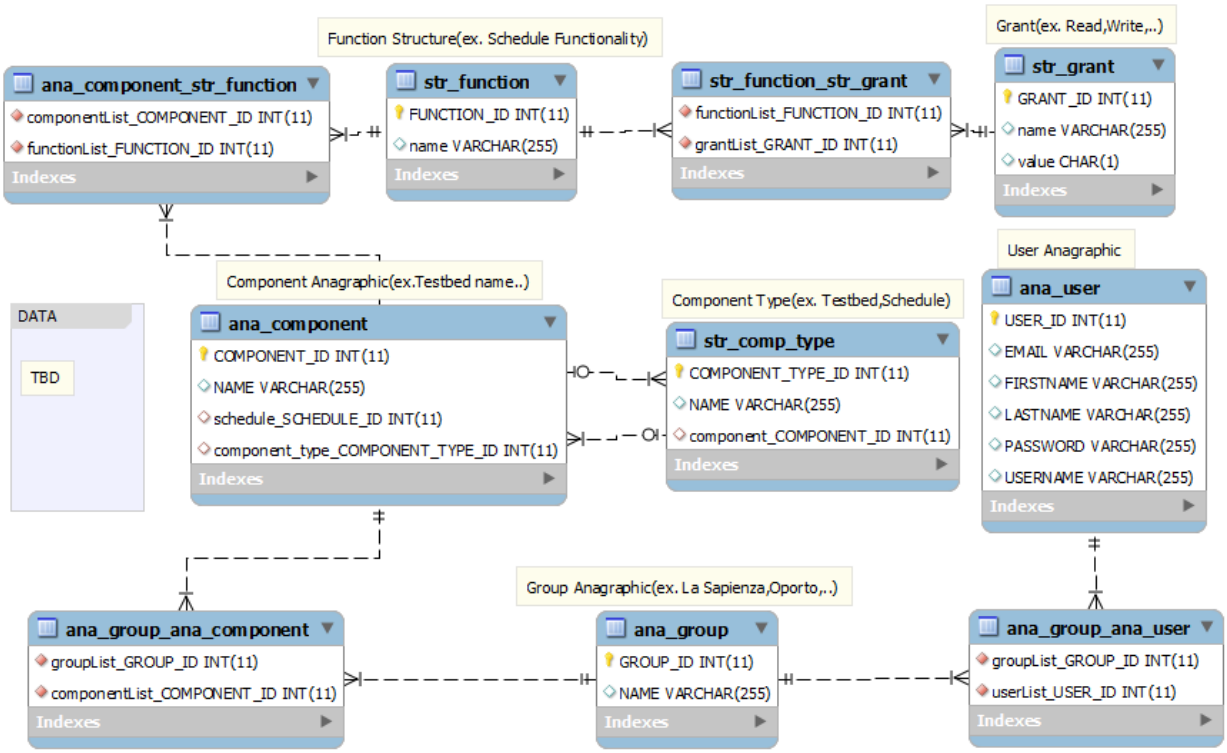


Figure 9