The Open-Multinet Upper Ontology

Towards the Semantic-based Management of Federated Infrastructures

Willner, A.; Papagianni, C.; Giatili, M.; Grosso, P.; Morsey, M.; Al-Hazmi, Y.; Baldin, I.

DOI
10.4108/icst.tridentcom.2015.259750

Publication date
2015

Document Version
Final published version

Published in
EAI Endorsed Transactions on Scalable Information Systems

License
CC BY

Citation for published version (APA):
The Open-Multinet Upper Ontology
Towards the Semantic-based Management of Federated Infrastructures

Alexander Willner
Technische Universität Berlin
Berlin, Germany
alexander.willner@tu-berlin.de

Chrysa Papagianni
Mary Giatili
NTUA, Athens, Greece
chrisap@noc.ntua.gr
mgiatili@netmode.ntua.gr

Paola Grosso,
Mohamed Morsey
University of Amsterdam
Amsterdam, Netherlands
{p.grosso|m.morsey}@uva.nl

Yahya Al-Hazmi
Technische Universität Berlin
Berlin, Germany
yahya.al-hazmi@tu-berlin.de

Ilya Baldin
RENCI/UNC
Chapel Hill, NC, USA
ibaldin@renci.org

ABSTRACT
The Internet remains an unfinished work. There are several approaches to enhancing it that have been experimentally validated within federated testbed environments. To best gain scientific knowledge from these studies, reproducibility and automation are needed in all areas of the experiment life cycle. Within the GENI and FIRE context, several architectures and protocols have been developed for this purpose. However, a major open research issue remains, namely the description and discovery of the heterogeneous resources involved. To remedy this, we propose a semantic information model that can be used to allow declarative interoperability, build dependency graphs, validate requests, infer knowledge and conduct complex queries. The requirements for such an information model have been extracted from current international Future Internet research projects and the practicality of the model is being evaluated through initial implementations. The main outcome of this work is the definition of the Open-Multinet Upper Ontology and related sub-ontologies, which can be used to describe and manage federated infrastructures and their resources.

Categories and Subject Descriptors
H.2.1 [Information Systems]: DATABASE MANAGEMENT—Logical Design; C.2.4 [Computer Systems Organization]: COMPUTER-COMMUNICATION NETWORKS—Distributed Systems; D.2.12 [Software]: SOFTWARE ENGINEERING—Interoperability

General Terms
Management, Standardization, Languages

Keywords
ontology, testbed, federation, semantic, management

1. INTRODUCTION
According to Vinton G. Cerf, the Internet remains an unfinished work.[1] Under the umbrella of Future Internet (FI) research, Internet architecture and protocols are being constantly enhanced in order to meet new requirements. Given the complexity of the Internet, an important approach to validate these enhancements is experimental evaluation within federated testbeds. Two important initiatives in this context are the Global Environment for Network Innovations¹ (GENI) in the United States (US) and the Future Internet Research and Experimentation² (FIRE) program in the European Union (EU).

To gain scientific knowledge from these experiments, control frameworks are needed to support the experimental life cycle in an automated and reproducible manner. This includes authentication, authorization, resource description, discovery, reservation, orchestration, provisioning, monitoring, and release, as well as experiment control and measurement[2]. Within a federated environment, these procedures become even more complex. Given the heterogeneity of the resources on offer in the testbeds, one particular issue that emerges is the description of these offerings. Currently, XML-based GENI Resource Specifications (RSpecs) with arbitrary extensions are being used to meet this objective. However, such a tree-based data structure does not define explicit semantics and therefore aggravates interoperability within a federation rather than enhancing it. As noted in [3], the simple union of two tree structures is not a tree anymore. Even if two XML files refer to the same resource, the related information is likely to be placed in different locations in the tree and additional choices must be made to obtain a new well-formed XML document.

One possible approach to solve this issue is the adoption of Semantic Web³ mechanisms. Without involving functional

¹http://geni.net
²http://ict-fire.eu
code, this would allow a platform to (i) semantically relate different resources and descriptions, (ii) detect errors in the models early and explain them, (iii) merge descriptions by expressing semantic information such as equality of resources, and (iv) conduct complex queries to discover resources that match specific requirements. Notable work that follows this approach in related contexts includes the Network Mark-Up Language[5] (NML), used to describe computer networks; the Network Description Language based on the Web Ontology Language[6] (NDL-OWL) for tested management; the Infrastructure and Network Description Language[7] (INDL) for modeling computing infrastructures; and the Networking innovations Over Virtualized Infrastructures[8] (NOVI) ontologies, with an additional focus on monitoring and policy management of federated environments.

We assume that the semantic description of resources is crucial to supporting interoperability between federated infrastructures. This includes not only FI testbeds, but also federated infrastructures in related fields, such as the Inter-cloud[9], the Internet of Things (IoT) or federated Software Defined Networks (SDNs). Therefore, a semantic model is needed that is not limited to a specific use-case.

Such a semantic model, which is re-usable for different fields of application, is the main contribution of the work presented here. The proposed Open-Multinet[10] (OMN) upper ontology re-uses and links to existing work that focuses on specific subdomains of the problem. While the work has mainly been validated within the FI experimentation context, its usefulness goes beyond this area.

The remainder of the paper is structured as follows. In Sec. 2 we discuss related work, before giving an overview of our approach and discussing some design decisions in Sec. 3. An evaluation of the approach in presented in Sec. 4. Finally, we give conclusions and an outlook in Sec. 5.

2. RELATED WORK

In order to place the contribution of the paper in context and identify the gap the work is intended to fill, we provide a short literature survey. This survey focuses on information modeling in general and the field of federated resource management in particular.

XML-based RSpecs are the accepted standard for describing the resource life cycle in GENI. These use a set of base XML-schemas and a number of extensions defined for specific purposes (e.g. stitching, OpenFlow and others). The original idea of RSpecs relied on structure-implied semantics in order to express facts about the resources. The location of the particular element or attribute within the Document Object Model (DOM) dictated its meaning. The extension mechanism was introduced in order to support the wider range of resources being incorporated into GENI. An extension is accomplished by using the XML <any> tag, which allows a chunk of XML conforming to the extension schema to appear virtually anywhere in the RSpec document. The upside of this approach is the infinite extensibility of RSpecs. The downside is the loss of structure-implied semantics for the extensions, since there is no mechanism for dictating which extension is allowed to appear in which part of the DOM. Syntax checking cannot be applied, since an RSpec document with a misplaced extension is typically syntactically correct. Instead, this leaves the checking process to the code parsing. This solution that does not scale in the long run and forces constraints on extensions to be expressed in procedural code, rather than declaratively, like the rest of the schema.

To overcome these issues and allow mutual understanding and minimum interoperation, a formal canonical reference model has to be introduced. As shown in Fig. 1, the current RSpec approach is to specify a loosely defined tree-based Data Model (DM) that is serialized in XML. After translation to an Object Model (OM), the contained information is validated by functional code. Encoding the information about provided, requested, controlled and monitored resources in a Semantic Model (SM) would allow us to exploit the advantages sketched in the introduction in a declarative manner. For these purposes, particular developments from within the Semantic Web[4] community could be adopted, namely the Resource Description Framework[11] (RDF), the Resource Description Framework Schema[12] (RDFS), the Web Ontology Language[13] (OWL), and the SPARQL Protocol And RDF Query Language[14] (SPARQL).

A number of fields of application have already adopted similar mechanisms. For example, search engines companies Bing, Google, and Yahoo! have collaborated to provide a vocabulary called Schema.org[15]. It provides a shared collection of thematic schemas used to annotate websites in a common way to allow search engines to recognize, evaluate and display their semantics. Further, within the federated cloud context, the Open-Source API and Platform for Multiple Clouds[16] (mOSAIC) ontology has been defined and further been adopted by the IEEE Standard for Intercloud Interoperability and Federation[17] (P2302). Other examples include the Machine-To-Machine Communication (M2M) community, which is developing SMs within the OneM2M Working Group 5 Management, Abstraction and Semantics[18] (MAS); and the semantic web services that have been developed under the Semantic Annotations for WSDL and XML Schema[19] (SAWSDL) umbrella. In fields related to federated testbeds, a variety of existing work has been defined, including NML, INDL, NDL-OWL and NOVI.

Based on the preliminary work of the Network Description Language[20] (NDL), NML is an information model designed to describe and define computer networks. The model underwent a thorough review and definition process to finally become an Open Grid Forum (OGF) standard. The developers of NML kept it as general as possible, with the possibility of extension in order to customize it for emerging network architectures and novel use cases.

INDL describes computing infrastructures in a technology independent manner. INDL also imports NML, which enables it to seamlessly include the networking part of a computing infrastructure. This ontology adds concepts and relations that are specific to the computing, processing and storage

4http://schema.org
5http://standards.ieee.org/develop/project/2302.html
6http://www.onem2m.org/mashome.cfm
part of an infrastructure, e.g. ProcessingComponent, and MemoryComponent. INDL further addresses the modeling of resource and service virtualization; it supports description, discovery, modeling, composition, and monitoring of resources.

NDL-OWL was one of the first attempts to design a resource life cycle ontology for GENI. Created to support the control framework Open Resource Control Architecture\cite{20} (ORCA), it grew out of the original NDL, however was extended in a number of important ways. The notion of resources and their life cycle was added into the ontology by creating the request, advertisement and manifest models. The models express information about a slice request, about the state of the substrate of the provider or the slice as built, respectively. Importantly, NDL-OWL supports the concept of multiple abstract delegation models \cite{21}, which can be constructed from a single detailed resource description of the substrate generated by the provider. The reason for the models is the need to preserve the privacy of the provider, allowing it to disclose only certain details of its internal topology. Several levels of abstraction were defined for NDL-OWL advertisements, with the switch being the most commonly used today. Within a switch, a single domain is abstracted into a switch fabric with multiple interfaces facing its peers. This abstraction succeeds in supporting inter-domain pathfinding. For more sophisticated topology embedding tasks, more detailed abstract models can be used.

The NOVI information model defines the semantics needed to describe resources and services, policy-based management systems and monitoring capabilities. It further describes communications in the NOVI architecture that focus on the federation of virtualized e-infrastructures. The NOVI information model enables semantic interoperation among the various software components of its architecture. The development of NOVI was driven by requirements including, in particular, the need to support virtualization concepts, context-aware resource selection, and harmonization of monitoring information and measurement units. As a result, the NOVI model comprises three main ontologies: resource, policy and monitoring. NOVI further imports NML for supporting network description. Although it is modular, vendor-independent, and uses the OWL language, the information model is limited to the scope of NOVI architecture. For instance, the resource ontology describes resources in NOVI by differentiating between physical and virtual nodes and network connectivity elements. Thus, extending this ontology to include resources from other domains (e.g. Wi-Fi, IoT) may not be straightforward.

The above overview of the current state of the art in describing resources within federated infrastructures has identified some of the main areas of focus and some drawbacks. A major outstanding problem is that the integration of each approach into current GENI and FIRE platforms is missing and a broader scope of application has not been considered.

3. OPEN-MULTINET ONTOLOGIES

A number of methodologies for ontology engineering already exist\cite{22, 23, 24}. In general, the purpose of the ontology should first be identified. The ontology can then be specified by capturing relevant concepts and relationships, and by integrating existing ontologies. Finally, after the information model has been formally encoded, the work can further be evaluated and documented. The purpose of our OMN upper ontology is comparable to the IEEE Standard Upper Ontology (P1600.1). Our ontology describes federated infrastructures and resources as generally as possible, while still supporting the management of their life cycle in federated environments. In the next section, the various parts of the ontology engineering process are outlined.

3.1 Design Decisions

Figure 2 is based on preliminary work called the Federated Infrastructure Description and Discovery Language\cite{25} (FIDDLE) and highlights the two perspectives needed for developing such an ontology. First, the different layers of concepts and relationships that describe resources within the GENI and FIRE contexts are presented (boxes). These include abstract administrative views, as well as lower concepts and re-usable concepts from existing ontologies. Second, methods of integration are suggested at each level in the form of concrete protocols appropriate to the given context (on the right side). In addition to recommending protocols used within the Semantic Web, suggestions are given for usage and extension of Slice-based Federation Architecture\cite{26} (SFA) Aggregate Manager (AM) and Clearinghouse (CH) APIs for resource discovery and provisioning, the Federated Resource Control Protocol\cite{27} (FRCP) for experiment control, and the ORBIT Measurement Library\cite{28} (OML) related OML Measurement Stream Protocol\footnote{http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html} (OMSP) for resource and experiment monitoring.

3.2 Upper Ontologies

Based on Figure 2, the OWL encoded OMN ontology is split into a hierarchy of a number of different ontologies (cf. Figure 4). The omn ontology on the highest level defines basic concepts and properties, which are then re-used and specialized in the subjacent ontologies. Included at every level are (i) axioms, such as the disjointness of each class; (ii) links to concepts in existing ontologies, such as NML, INDL and NOVI (cf. Figure 3); and (iii) properties that have been shown to be needed in related ontologies.
The **omn-federation** ontology describes federations, along with their members and related infrastructures. Basically, a federation can have members (**omn-federation:hasFederation-Member**) and both federations and their members are individuals of type **schema:Organization**. Further, a **schema:Organization** can manage (**omn-federation:administers**) one or multiple testbeds (**omn-federation:Infrastructure**). This allows a high level entry point to be defined, from which the user of a federation can discover available resources, services and APIs.

The **omn-lifecycle** ontology describes the whole life cycle of resource/service management in the federation. This includes requests, reservation (schedule for allocation), provisioning and release. These stages of the life cycle are modeled using the following classes: (i) **Request**, which expresses a collection of resources and/or services and the relationships between them, as requested by an experimenter; (ii) **Confirmation**, which provides a collection of resources and/or services that are reserved and scheduled to be allocated at a future point in time; (iii) **Manifest**, which describes a collection of resources and/or services currently provisioned by the experimenter; and (iv) **Offering**, which provides all resources and services that can be allocated to a user’s request. These four objects are subclasses of **omn:Topology**. The description of a resource/service life cycle is complemented using classes that provides the current state of the resource/service in question (e.g. **Pending**,** Ready**, **Started**, **Stopped** etc). Finally the reservation of a Resource, Service or Topology is modeled using the **omn:Reservation** class. The object property **omn:hasReservation** expresses the relation between them, while **omn-lifecycle:ReservationState** denotes the state of reservation with regards to the Resources, Services or Topology (e.g. **Allocated**, **Provisioned** or **Unallocated**).
A resource in the OMN ontology is defined as any provisionable, controllable, and/or measurable entity. The resource ontology augments the definitions of the Resource class defined in the main OMN upper ontology. In other words, all concepts of this ontology are descendants of class Resource. The resource ontology has four fundamental concepts, namely NetworkObject, Node, Interface, and Link. The NetworkObject concept represents a generic resource object involved in networks, e.g., ports and links. The Node class describes a device that is connected to, or part of, a network, and does not necessarily represent a physical machine. Interface is a connectivity interface, which represents a physical or logical transport entity. Link class represents a unidirectional data transfer from its source to a sink of NetworkObjects. It is worth noting that the class NetworkObject is the direct ancestor of all other classes.

A component is defined in the OMN ontology as any entity that is part of a Resource or a Service, but which does not need to be a Resource or a Service itself. The Component ontology covers concepts that are considered descendants of the Component class defined in the OMN upper ontology. The Component ontology comprises several classes representing a set of fundamentals in the context of the OMN application field, e.g., CPU, Sensor, Core, Port, Image, etc. Any of these classes can be the range of the property hasComponent, whose domain can be a Resource, a Service or even another Component.

A service is defined in the OMN ontology as any entity that has an API to use it. A service may further depend on a Resource. The Service ontology covers different services in the relevant application areas, such as Aggregate Manager, Portal, Measurement Service, Hadoop, Broker, etc. Similar to OMN Service are novi:Service, nmni:Service and indl:Capability classes. Any OMN Service can be the range of the property hasService, whose domain can be a Group, a Resource, or even another Service.

The variety and heterogeneity of monitoring information within federated infrastructures results in fragmentation of the provided data with respect to their formats and representations. The Monitoring ontology is directly linked to other OMN ontologies and facilitates interoperability in terms of enabling common monitoring data to be exchanged federation wide. It is built based on existing ontologies, such as NOVI and Monitoring and Measurement in the Next generation Technologies[29] (MOMENT). A Metric class represents any entity that can be measured. It is defined in a generic way to cover the common measurement metrics in the relevant areas of application. It defines metrics whose information changes dynamically and metrics whose information may change very infrequently. The Data class describes the main concepts related to measurement data. The Unit class defines data units, covering both binary and decimal unit prefixes, and more. The Generic Concepts class represents general, global concepts (e.g. location).

Policies are rules governing the choices in behavior of a system[30]. Correct expression of policies and proper communication of their implications is a fundamental component in the creation of federations. Management and authorization mechanisms all need to be properly captured, and the OMN community is currently investigating the most appropriate way to model them. Previous work, for example in NOVI, with its policy ontology, shows that such an ontology should leverage the infrastructure and resource description, and couple this with specific policy concepts. Furthermore the fact that policies and authorizations have a very dynamic nature will require solid software tools to handle the information delivered by the policies.

The OMN ontology is designed in a flexible, extensible way to cover specific domains. Examples of domains of interest to OMN include wireless (e.g., Wi-Fi or sensors), SDN, and Cloud computing. OMN currently includes a domain ontology omn-wireless that describes wireless resources and capabilities. It includes classes representing, for example, omn-domain-wireless:WirelessInterface, Channel, and Frequency, as well as object properties like omn-domain-wireless:useFrequency, and data properties like omn-domain-wireless:channelNum, omn-domain-wireless:lowerBoundFrequency and omn-domain-wireless:upperBoundFrequency. Further domain specific ontologies are either under design (e.g. omn-domainsdn) or planned for design.

### 4. EVALUATION

The applicability of the proposed ontology has been validated within the FIRE context. More specifically, requirements from the Federation for FIRE[2] (Fed4FIRE) project have been incorporated and, as a result, a number of mappings between GENI RSpecs and the semantic model will be presented in this section. Further, details are given regarding how to model and use resource reservation and interconnection information for resource discovery and topology embedding.

#### 4.1 Automated Life Cycle Translation

Appropriate tools are needed to facilitate the transition of non-semantic management systems towards using graph based information models and the integration of semantic management systems into the GENI context. These tools should support translating locally used structured, semi-structured and unstructured data models into RDF-based data and the translation from RDF data into GENI RSpecs. This approach has several advantages. The tools (i) automate and speed up the process of converting non-RDF data; (ii) encourage users and developers to migrate their systems to using Semantic Web technologies; and (iii) ensure that the quality of generated RDF data corresponds to its counterpart data in the original system.

As a result, we have developed a translation tool to convert stateless GENI RSpec XML documents into RDF and back using the OMN ontology. Our tool parses the XML tree and converts the tags and attributes to their corresponding classes or properties; it also supports converting the complete GENI resource life cycle messages. To give a better understanding of this translation process, we provide some illustrative examples for conversions of Advertisements, Requests and Manifests.

The implementation of the translation tool follows a Test Driven Development (TDD) approach, is included in a Continuous Integration (CI) environment with test coverage analytics, and is offered as a Java based open source library.
("omnlib") in a public maven repository. It uses the Java Architecture for XML Binding (JAXB) and Apache Jena to map between XML and RDF and Java objects, and supports a number of APIs: (i) a native API to be included in other Java projects; (ii) a CLI to be used within other applications; and (iii) a REST based API to run as a web service.

The listings that follow serve two purposes. First, they present detailed and understandable examples on how to translate GENIRSpecs into OMN RDF graphs. Second, they demonstrate how to uniquely specify any kind of resource. While the default GENIRSpec mainly defines Nodes and Links, we adopted an example used in the documentation of the experiment control system cOntrol and Management Framework[31] (OMF): a Garage that manages a Motor. This example also sketches the possibility of using the same information model for a handover from resource provisioning to resource control frameworks (cf. Figure 4).

Advertisement. Listing 1 shows a simple GENI Advertisement RSpec used to publish available resources within a GENI federation. The example shows a single node of type MotorGarage that can provision the sliver type Motor. While traditionally hardware_type and sliver_type used to be simple strings, unique Uniform Resource Identifiers (URIs) are used here to provide machine interpretable information.

Listing 1: RSpec Advertisement (in)

```
<rspec
  xmlns="http://www.geni.net/resources/rspec/3"
  type="advertisement"
>
  <node
    component_manager_id="urn:publicid:IDN+testbed.example.org+authority+cm"
    component_id="http://testbed.example.org/resources/motorgarage-1"
    exclusive="false"
  >
    <sliver_type name="http://open-multinet.info/ontology/resources/motorgarage#MotorGarage"/>
    <available now="true"/>
    <location longitude="3.734761" latitude="51.036145"/>
  </node>
</rspec>
```

Listing 2 shows the converted graph, serialized in Turtle. The overall approach is to define an omn:Topology, here the subclass omn-lifecycle:Offering, that contains pointers to the offered resources. Each resource is an individual of a specific type that is parent of (i.e. can provision) one or more specific types. Other information, such as the location, is translated by re-using well-known existing ontologies.

Listing 2: OMN Offering

```
example:advertisement a omn-lifecycle:Offering ;
  omn:hasResource <http://testbed.example.org/resources/motorgarage-1> .

<http://testbed.example.org/resources/motorgarage-1> a motorgarage:MotorGarage ;
  omn:isResourceOf example:advertisement ;
```

In order to demonstrate a complete round trip translation, i.e. from XML to RDF and back to XML, the Advertisement RSpec is shown once more in Listing 3. Note that this example has been statelessly generated based solely on the graph from Listing 2. All information has been converted, making Listing 1 and Listing 3 equivalent expect for the component_manager_id, which is the subject of an ongoing discussion about how it should best be modeled.

Listing 3: RSpec Advertisement (out)

```
<rspec
generated="2015-02-12T09:46:59.480+01:00"
  generated_by="omnlib"
  expires="2015-02-12T09:46:59.480+01:00"
  type="advertisement"
  xmlns="http://www.geni.net/resources/rspec/3">
  <node
    component_id="http://testbed.example.org/resources/motorgarage-1"
    component_name="MotorGarage"
    exclusive="false"
  >
    <sliver_type name="http://open-multinet.info/ontology/resources/motorgarage#MotorGarage"/>
    <available now="true"/>
    <location longitude="3.734761" latitude="51.036145"/>
  </node>
</rspec>
```

Request. After receiving an Advertisement RSpec, the next step is to request a specific sub-topology. In Listing 4 such a simple Request RSpec is shown. Again URIs are used to specify the type of resource being requested from a specific node. In order to be able to map the requested resource to the provisioned resource at a later stage, the client_id string is set and converted.

Listing 4: RSpec Request (in)

```
<rspec
  xmlns="http://www.geni.net/resources/rspec/3"
  type="request">
  <node
    component_manager_id="urn:publicid:IDN+testbed.example.org+authority+cm"
    component_id="http://testbed.example.org/resources/motorgarage-1"
    client_id="myMotor"
  >
    <sliver_type name="http://open-multinet.info/ontology/resources/motor#Motor"/>
  </node>
</rspec>
```

The conversion process is again shown in Listing 5. An omn:Topology, here an omn-lifecycle:Request, has been created with pointers to the requested resources. The resource is an individual of the requested type that is implemented by a specific resource and has the above mentioned identifier. Note that the property implemented_by is only set if the request contains information about where a sliver should be
created, i.e. in case of a bound request. Otherwise an unbound request has been sent that has to be processed further and enhanced by a resource mapping mechanism.

**Listing 5: OMN Request**

```plaintext
1 example:Request a omn-lifecycle:Request ;
2 omn:hasResource example:myMotor .
3 example:myMotor a omn:lifecycle:Motor ;
4 omn:lifecycle:hasID "myMotor" ;
5 omn:lifecycle:implementedBy <http://testbed.example.org/resources/motorgarage-1> .
```

**Manifest.** After successfully provisioning the requested resources, a manifest is returned. As shown in Listing 6 an omn:Topology, here an omn-lifecycle:Manifest, has once again been defined to identify the provisioned resources. The relevant individual is of the requested type, is identified further with the client identifier and is implemented/provisioned by a specific resource.

**Listing 6: OMN Manifest**

```plaintext
1 example:manifest a omn-lifecycle:Manifest ;
2 omn:hasResource <http://testbed.example.org/motorgarage-1/motor-1> ;
3 <http://testbed.example.org/motorgarage-1/motor-1> a omn:lifecycle:Motor ;
4 omn:lifecycle:hasID "myMotor" ;
5 omn:lifecycle:implementedBy <http://testbed.example.org/resources/motorgarage-1> .
```

The translation into a GENI Manifest RSpec is shown in Listing 7. Note that the client_id identifies the requested resource and the sliver_id, the unique identifier of the newly created resource instance within the testbed, follows the GENI standard with implied semantics within a simple string. However, a URI is again used after the last “+” sign to allow internal semantic management of the resource without relying on GENI-related notions.

**Listing 7: RSpec Manifest (out)**

```plaintext
1 <rspec
generated="2015-02-12T09:41:25.230+01:00"
generated_by="#omnlib"
type="manifest"
xmlns="http://www.geni.net/resources/rspec/3">
2 <node
3  client_id="myMotor"
4  component_id="http://testbed.example.org/resources/motorgarage-1"
5  component_name="motorgarage-1"
6  sliver_id="urn:publicid:TB0+testbed.example.org+sliver=http://empusa.ipv6.lip6.fr/ontology/resources/motor#Motor"/>
7 </node>
8 </rspec>
```

**4.2 Performance Assessment**

Besides the functional principle of the converter, its performance is of further interest. The input of the following performance evaluation is based on the RSpec Advertisement published by the Virtual Wall testbed, whose XML serialization is about 2.4 MB in size. In total 212 nodes, including their 619 sliver and 1297 hardware types, were translated.

The evaluation is divided into two parts. The first part includes the conversion of the XML document into a JAXB OM. The measurements were repeated 100 times with 1 second breaks in between and 10 repetitions were executed before filtering out possible start up, initialization and compilation outliers. This conversion takes with a 95% confidence interval, 5263 ms +/- 15 ms. As a result, this finding is left out of the following visualization and should further be examined and, if possible, optimized.

The second part includes the conversion process between the JAXB OM, the RDF graph and the serialization back to XML. The measurements were repeated 1000 times with 100 ms breaks in between and 10 warm-up repetitions. As shown in Figure 5, the most expensive operation is the XML serialization, and the mapping between the OM and the RDF tree takes about 6 ms.

![Figure 5: Performance of the conversion process](diagram)

**4.3 Resource Description and Reservation**

The description of the functional properties of resources and services is necessary to facilitate resource discovery and mapping. An offering provides the description of a physical infrastructure that could be regarded as a set of resources available to be allocated to an experimenter’s slice. It is modeled as an omn:Topology (a subclass of omn:Group).

An example of an omn-lifecycle:Offering based on the PlanetLab Europe Advertisement RSpec is depicted in Figure 6. For the sake of readability, only a subset of the advertised resources is depicted i.e. two nodes and their two corresponding interfaces, and a part of their advertised attributes. All resources are modeled using the omn:Resource class. For example, nodes planetlab2.thlab.net and empusa.ipv6.lip6.fr are modeled as individuals of the subclass Node. The property omn-resource:isExclusive (Boolean flag), indicates whether or not the physical resource can be assigned exclusively to an experimenter’s slice. Each resource must have a unique identifier expressed by the data property omn-lifecycle:hasID.
To facilitate the association with an Advertisement RSpec, the URI included in the corresponding RSpec’s component_id is used as a unique identifier.

Each resource may be reserved for one or more periods of time. Specifically, for resources that may be exclusively reserved by experimenters, availability should be explicitly advertised. Adopting the OMN Information Model (IM), the reservation of a resource is modeled using the omn:Reservation class. The reservation object is used to describe the lifetime of resource provisioning i.e. when the reservation starts and when it ends. Lifetime is modeled using the interval class of the time ontology, defined as subclass of the Reservation class. In Figure 6, resources emp.ua ipv6.ip6.fr and node1.1953:eth0 are reserved at the time of issuing the omn-lifecycle:Offering (being in a provisioned state). The reservation instance of the rest of the resources refers to reservation in the future (allocated state). Each resource may have multiple reservations, depending on their past, current and future reservation state.

4.4 Discovery and Topology Embedding

The problem of mapping requests for virtual networks to specific nodes and paths in a physical network in the context of network virtualization is commonly referred to as Virtual Network Embedding or Topology Embedding[32]. In the context of federated testbeds, topology embedding describes the process of mapping requested resource/topologies to the federated environment.

Exploiting semantic annotations for testbed infrastructure allows precise control of the mapping process between requested and available resources. Such mapping at the desired level of abstraction is important for supporting user requests and adjusting operational objectives. Moreover, one of the main advantages of RDF, making it suitable for describing networks, is that its data constitutes a directed graph. Based on this graph, the path between source and sink node(s) can be calculated; this bears similarity to the real network the graph represents.

The OMN translation tool supports extracting the network topology information and converting it into RDF. Listing 8 indicates a sample RSpec request with two nodes, each one with an interface and a link connecting the two nodes together. The translation tool converts the RSpec request into RDF triples as indicated in Listing 9. SPARQL queries can be utilized to check if there is a path between two nodes, and return such a path if it exists.

<table>
<thead>
<tr>
<th>Listing 8: Example RSpec request with networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rspec type=&quot;request&quot; xmlns=&quot;http://www.geni.net/resources/rspec/3&quot;&gt;</td>
</tr>
<tr>
<td>&lt;node client_id=&quot;geni1&quot; component_manager_id=&quot;urn:publicId:IDN+uncvmsite+authority+cm&quot;&gt;</td>
</tr>
<tr>
<td>&lt;address ip_address=&quot;172.16.22.1&quot; netmask=&quot;255.255.255.0&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/node&gt;</td>
</tr>
<tr>
<td>&lt;node client_id=&quot;geni2&quot; component_manager_id=&quot;urn:publicId:IDN+uncvmsite+authority+cm&quot;&gt;</td>
</tr>
<tr>
<td>&lt;address ip_address=&quot;172.16.22.2&quot; netmask=&quot;255.255.255.0&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/node&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 9: RSpec request converted into RDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>example:request a omn-lifecycle:Request ;</td>
</tr>
<tr>
<td>omn:hasResource example:geni1, example:geni2 .</td>
</tr>
<tr>
<td>example:geni1 a example:m1.small, omn-resource:Node ;</td>
</tr>
<tr>
<td>omn-resource:isSink example:center ;</td>
</tr>
<tr>
<td>omn-resource:isSource example:center .</td>
</tr>
<tr>
<td>example:geni2 a example:m1.small, omn-resource:Node ;</td>
</tr>
<tr>
<td>omn-resource:isSink example:center ;</td>
</tr>
<tr>
<td>omn-resource:isSource example:center .</td>
</tr>
<tr>
<td>example:geni1 a example:geni2 .</td>
</tr>
<tr>
<td>example:geni10 a example:request .</td>
</tr>
<tr>
<td>example:geni1:0 a example:Interface .</td>
</tr>
<tr>
<td>example:geni2:0 a example:Interface .</td>
</tr>
<tr>
<td>example:request a omn-lifecycle:Request .</td>
</tr>
<tr>
<td>example:geni1 a example:geni2 .</td>
</tr>
<tr>
<td>example:geni1:0 a example:Interface .</td>
</tr>
<tr>
<td>example:geni2:0 a example:Interface .</td>
</tr>
<tr>
<td>example:geni1 a example:geni2 .</td>
</tr>
<tr>
<td>example:geni10 a example:request .</td>
</tr>
</tbody>
</table>

Overall, topology embedding SPARQL queries can be constructed dynamically based on the request[33]. For example a user may request exclusive access for two hours to a computing node within a 5km radius distance from the center of Paris, starting from the time the request is issued.

The corresponding SPARQL query (cf. Listing 10) is provided based on the offering shown in Figure 6. In PlanetLab, reservable computing nodes can be allocated to a single slice at any given time. In the example, the experimenter needs to reserve such nodes for specific time slots. The SELECT part of the query identifies as a variable a single Node instance with the data property omn-resource:isExclusive set to true. The reservation time slot and the location requirement are considered constraints imposed on the mapping problem. Therefore all computing nodes that map onto the requested properties (e.g., exclusive access, within a 5 km radius) and are not already reserved during the requested period defined in the FILTER are considered possible candidates for allocation.

<table>
<thead>
<tr>
<th>Listing 10: SPARQL Mapping Query Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT ?resource WHERE {</td>
</tr>
<tr>
<td>?resource omn-resource:isExclusive &quot;true&quot; &quot;xsd:boolean&quot; .</td>
</tr>
<tr>
<td>filter( (?lat &gt; 48.858222-xsd:float(?lat)) &amp;&amp; (?lat &lt; 48.858222-xsd:float(?lat)) &amp;&amp; (?lon &gt; -0.155534875-xsd:float(?lon)) &amp;&amp; (?lon &lt; -0.005003701-xsd:float(?lon)) )</td>
</tr>
<tr>
<td>MINUS {</td>
</tr>
<tr>
<td>SELECT ?resource WHERE {</td>
</tr>
<tr>
<td>?resource omn-resource:isExclusive &quot;true&quot; &quot;xsd:boolean&quot; .</td>
</tr>
<tr>
<td>filter( (?lat &gt; 48.858222-xsd:float(?lat)) &amp;&amp; (?lat &lt; 48.858222-xsd:float(?lat)) &amp;&amp; (?lon &gt; -0.155534875-xsd:float(?lon)) &amp;&amp; (?lon &lt; -0.005003701-xsd:float(?lon)) )</td>
</tr>
<tr>
<td>} }</td>
</tr>
</tbody>
</table>

Example SPARQL query for filtering candidates for allocation.
Restricting the LIMIT to 1 would provide a node mapping solution, in this case planetlab2.thlab.net, as shown in Listing 11.

Listing 11: SPARQL Mapping Query Result

```
$ sparql --repeat=10,1 --time --data=offering.ttl --query=query.sparql

---------------------------------
<table>
<thead>
<tr>
<th>resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>----------</td>
</tr>
</tbody>
</table>

Time: 0.163 sec
```

5. CONCLUSION AND FUTURE WORK

We have given an overview of the important issue of describing resources within federated infrastructures. Motivated by the concrete field of application of experimental Future Internet research, we have further presented related work on this topic.

The crucial results are twofold. First, we identified that mechanisms developed within the Semantic Web present promising means to address this issue. Second, based on, and integrated with, existing work in the field, we have developed and demonstrated the Open-Multinet Upper Ontology.

The presented work provides potential advantages for infrastructure owners, federation operators, developers, and users. While owners and operators can offer highly heterogeneous resources by specializing existing concepts, developers and users have the possibility to conduct complex queries to discover them. Within a federation, it is possible to enhance this process even further by relating offered descriptions with each other by expressing e.g. equality of resources. Tool developers can re-use existing work available within the Semantic Web to easily explain errors to users, allow handovers between protocols or to implement complex resource matching or path finding algorithms without involving functional code.

Our short-term goal is to include support for our ontology in SFA AMs like FITeagle and SFA user tools such as jFed. In the medium term, we want to broaden our approach to include further tools for resource scheduling, experiment control and monitoring. The long-term goals include utilizing our approach in further fields of application, such as federated cloud environments. Towards this goal, we have already extended the translation tool to support the Organization for the Advancement of Structured Information Standards (OASIS), specified by the Topology and Orchestration Specification for Cloud Applications (TOSCA).

Acknowledgments

Research for this paper was partially financed by the European Union’s FP7 grant agreement no. 318389 Fed4FIRE Project, and the Dutch national program COMMIT. We...
thank our project partners for their contributions and their collaboration on this research work.

References