Managing Federations of Virtualized Infrastructures: A Semantic-Aware Policy Based Approach

Leonidas Lymberopoulos*, Paola Grosso†, Chrysa Papagianni *, Dimitrios Kalogeras *, George Androulidakis *,
Jeroen van der Ham†, Cees de Laat †, and Vasilis Maglaris*
**Network Management & Optimal Design Laboratory - NETMODE, School of Electrical & Computer Engineering*
National Technical University of Athens - NTUA, Zografou, Greece
Email: {leonidas,chrisap,gandr,dkalo,maglaris@netmode.ntua.gr}
†*SNE (System and Network Engineering Group)*
Universiteit van Amsterdam - UvA, Netherlands
Email: {p.grosso,vdham,delaat@uva.nl}

***Abstract*—This paper presents our work toward organizing and managing various forms of federations of virtualized infrastructures. We adopt the Ponder2 policy framework and the SMC architecture as a powerful engineering approach, which we apply to semantic-aware management of federations of Future Internet (FI) virtualized infrastructures. To cater for context-awareness, we plan for a common information model, based on the Network Description Language (NDL), capturing a common set of abstractions of virtualized resources and services, nodes, routers and switches, custom network topologies with specic bandwidth demands, etc.**

To handle management of generic complex federated environments, we employ structural patterns to model federations as graphs, whose vertices represent SMCs and edges denote the type of relationship between them. We give an illustration of such structures corresponding to existing FI experimental platforms in the US and Europe and we provide examples containing inter-domain management responsibilities as missions. Finally, we propose to augment the Ponder2 framework with single & multi-domain resource provisioning capabilities, enabling efficient sharing of virtualized networked facilities among federation users.

***Keywords*-Virtualisation platforms; Federated Experimental Infrastructures; Network Description Language - NDL, Policy Based Network Management - Ponder2 ; Self Managed-Cell - SMC;**

I. Introduction

Over the last years, a number of experimental infrastructures have been deployed to provide the networking research community with the networking and computing facilities necessary to test and validate Future Internet (FI) protocols, architectures and applications. Important features of FI infrastructures include the implementation of virtualization techniques for sharing resources and services and support for federation mechanisms to address scalability issues and enable interoperability across heterogeneous platforms. Examples include the PlanetLab [6], VINI [7] and Emulab [8] infrastructures in the US (GENI NSF initiative [4]) and the Panlab [9], OneLab [11] and FEDERICA [10] infrastructures in Europe (FIRE EC initiative [5]).

The rationale behind a federation is the ability to allow experimenters to use baskets of heterogeneous resources drawn from individual participating infrastructures. This way, a user can access a wide range of services and resources *e.g.* computing and storage end-nodes, wireless and sensor nodes, routers and switches, custom network topologies with specific bandwidth demands etc. Indeed, federation reflects the foreseen model of the Future Internet as a universal environment capable of providing seamless connectivity mechanisms to heterogeneous networked devices.

Current federation efforts concentrate on development of tools and APIs to federate virtualized infrastructures for research and experimentation for the Future Internet. Example include the bottom-up federation proposed in the Slice Federation Architecture (SFA) [2], widely adopted in US (GENI) and in Europe (OneLab), and the top-down federation via the early GENI Clearinghouse [13] plans and the Teagle [3] architecture in the European Panlab testbed.

In this paper, we address the problem of how to engineer federations into various structures, ranging from hierarchical to purely decentralized ones. We employ a flexible engineering approach for establishing relations between different virtualized infrastructures allowing infrastructure providers to establish various forms of provider-to-provider agreements.

We provide a solution to the problem of engineering of federations using some concepts introduced in the Ponder2 Policy framework [15], which has been developed for engineering of pervasive computing systems. We extend the Self- Managed Cell (SMC) architecture [16] of Ponder2's framework with provisioning mechanisms to enable efficient sharing of virtualized networked facilities among federation users.

To provide interoperable mechanisms for managing federations of heterogeneous virtualized infrastructures, we employ a common information model that captures the concepts and the semantics of resources and services offered by several virtualization platforms, focusing initially on the FEDERICA and the PlanetLab platforms. Our information

model is based on the Network Description Language (NDL) [23], [25], that enables the development of intelligent context-aware methods and algorithms. The corresponding data model could

This work is motivated by ongoing research in the FIRE FP7 STREP project *NOVI - Network Innovation over Virtualized Infrastructures* [1]. NOVI's research concentrates on methods and algorithms to compose virtualized e-Infrastructures towards a holistic Future Internet (FI) cloud service. In this paper, we report directions consistent with NOVI's vision, *i.e.* to devise, develop and validate mechanisms for policy-based control and management. These are leveraged with context-aware discovery and provisioning of shared resources and services within federations of heterogeneous infrastructures.

The paper is organized as follows. Section II presents the Ponder2 Policy framework that we use in our work. Section III presents an NDL-based domain-independent data model that captures the main abstractions of shared resources and services. Section IV discusses how we can structure and manage various forms of federations and in Section V we illustrate how resource provisioning can be realized within a complex federation structure, whose entities are described in our NDL-based data model. Section VI presents an overview of related work for federating virtualized infrastructures. Finally, section VII provides a summary and conclusions of this paper.

II. Background Work: Ponder2 Policy Framework

Ponder2 implements the Self-Managed Cell (SMC) architecture [16], as shown in Figure 1. Management services interact with each other through asynchronous events propagated through a content-based event bus. The discovery service, policy service and event-bus constitute the core functionality of the SMC and must always be present in the implementation of a SMC. More information on the SMC architecture and the Ponder2 policy language used for defining policies within a SMC can be found in [14] and [15] respectively.

The SMC architecture has been proposed in [14] as an engineering paradigm for structuring ubiquitous systems. It provides a well-defined interface for interaction with other SMCs and can be tailored and deployed on various environments at different levels of scale, from sensor networks to large distributed systems. For the purposes of our work, we are primarily focusing on the *structural* patterns that enable establishment of inter-SMC relations within large-scale federated distributed systems. Detailed information on all supported patterns and their behaviour can be found in [14]. We will here describe the structural patterns which we use in our work as a means to engineer various federation structures.

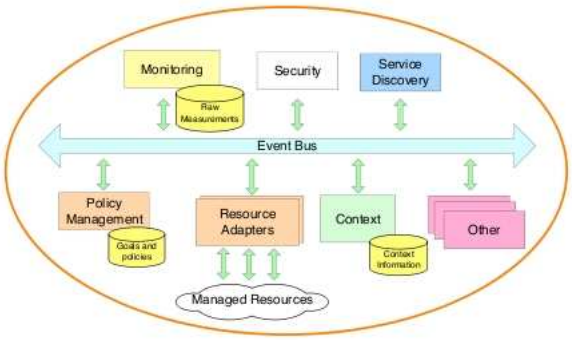

Figure 1. Ponder2 Self Managed Cell Architecture

Structural patterns define how SMCs can establish relationships with other SMCs. Patterns of this type are the following:

- Composition: Whereby an outer SMC encapsulates a number of inner SMCs, the latter being its internal resources. Any interactions to the inner SMCs are realized through the outer SMC.
- Peer to Peer: In this pattern, SMCs can interact between each other and act as equal entities, following the Peer-to-Peer network communications paradigm.
- Aggregation: This pattern reflects a hierarchical relationship where a number of "lower-level" SMCs provide their services to "higher-layer" SMCs, which can offer to clients baskets of services and resources, drawn from the "lower-level SMCs".

III. Network Description Language Extensions for Virtualized Infrastructures

We adopted the Semantic Web approach [24] for describing the semantics of shared resources and services within virtualized infrastructures. Our decision was driven by the need to devise context-aware algorithms for resource discovery/composition and intelligent decision making. Management decisions in an FI environment are influenced by context. For example, resource utilization and constraints dictate how provisioning of user-requested virtual resources should be embedded within a shared substrate, as we will discuss in Section V. This information can be retrieved by monitoring systems and stored in semantically rich structures - ontologies.

Our work builds upon the Network Description Language (NDL) [23] that provides a classification of terms and concepts related to computer networks in the form of several RDF schemas. To capture all the concepts and terms required for management and control of virtualized platforms, we created an extension schema that links to existing NDL schemas using parent classes and/or properties.

NDL's current version defines a *domain schema*, which is a *vocabulary* for describing administrative network domains

and abstracted views of the devices within them. Figure 2 shows the RDF classes and predicates currently defined in NDL's *domain schema*.

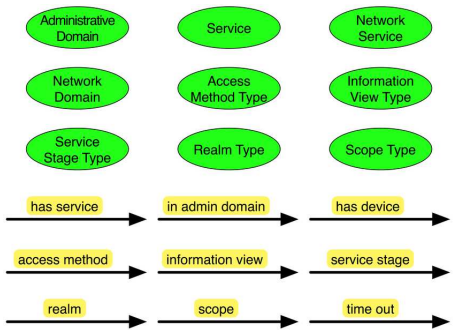

Figure 2. The NDL domain schema, with the RDF classes and RDF properties defined

In the above schema, the *Network Domain* class is used to represent a collection of network elements behaving as a *black box* that offers external interfaces. A *Network Domain* is abstracted as an aggregation of different devices without the need to specify the internal details of constituent devices.

Figure 3 shows indicative RDF classes of our extension schema, illustrating their relation to existing NDL classes.

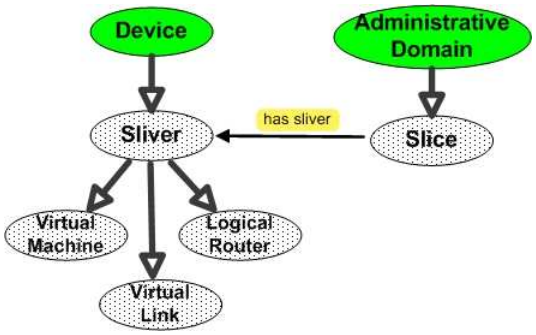

Figure 3. Indicative classes of the NDL extension schema

We define the *Sliver* class to denote single virtual resources and the *Slice* class as a collection of virtual resources allocated to user-requests for running specific applications and services within the federated environment. As shown in Figure 3:

1) *Sliver*. A *Sliver* class is a subclass of the *Device* class, which is defined in the *Topology* schema of NDL.
2) *Slice*. A *Slice* is a subclass of the *Administrative Domain* class. The *Slice* class is an aggregation of *Sliver* entities.

As shown in Figure 3, we further refine the Sliver class to represent virtual entities within current virtualized infrastructures, which provide virtualization either at the level of the Operating System (O/S) or at the networking layer, or both. The Slivers which we currently introduced were selected among entities within existing FI experimentation facilities; namely, PlanetLab, FEDERICA and VINI.To address the problem with Sliver as a subclass of Device, note that a virtual resource will host another virtual resource, which is not directly possible if we made a distinction between Logical and Physical resources.

Indeed, in the NML [32] group there have been discussion and the decision was making a distinction between Physical and Virtual resources. Mostly because it is not always possible or desirable to make that distinction.

1) *Virtual Machine*. A *Virtual Machine - VM* describes a virtual server instantiated on a physical node. This could be a PlanetLab VM or a FEDERICA VM, and does not depend on specific virtualization technology (e.g. hypervisor S/W).
2) *Logical Router*. A *Logical Router* describes an entity providing routing functionality either as a result of hardware virtualization (e.g. a FEDERICA Juniper MX-series logical router) or it can be a routing entity implemented in software (e.g. a Click modular S/W running within a User Mode Linux - UML VM in the early VINI implementation [22]).
3) *Virtual Link*. A *Virtual Link* describes a link connecting either two instances of the *Logical Router* class, or two physical L2 *Switches* (as defined in the existing NDL schemas). It can be implemented as MPLS paths, or tunnels (e.g. IP over GRE, L2TPv3, Ethernet over GRE) between *Logical Router* entities or VLANs between physical L2 *Switches*. For example, implementation within the current VINI platform is done through an Ethernet over GRE tunnel between two routing entities [26]. In FEDERICA, VLAN technology is used to link at L2 two FEDERICA Juniper Logical Routers or physical switches, connected by SDH/SONET 1Gbps circuits [10].

IV. Modeling Federations using the SMC Structural Patterns

In this section, we present a methodology for describing complex federation structures in FI environments employing *structural* patterns of the SMC architecture. Figure 4 presents a graph model of a federation in which vertices are SMCs representing virtualized infrastructures. The latter can be viewed as autonomous domains, implementing their own management policies and control plane tools. Directed arrows represent the *structural pattern* that defines how an SMC establishes a relationship with a neighbor SMC.

In the above Figure, we depict a federation of six (6) different virtualized infrastructures (platforms), each one represented by a single SMC. The overall federation is structured in a way to reflect platform relationships defined as structural patterns of the SMC architecture (composite, peer-to-peer and aggregation).

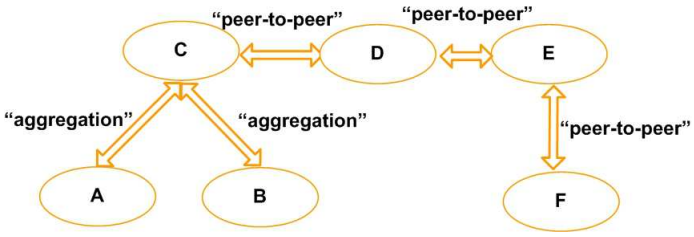

Figure 4. A graph model of a federation structure

In Figure 4, A and B reflect lower-layer SMCs (e.g. local testbeds) related using the *aggregation* pattern to C. For example, C could be the European Panlab [9] virtualization infrastructure. Note that Panlab's Teagle architecture [3] is implemented following this hierarchical concept. C interacts with D using the *peer-to-peer* pattern (e.g. Panlab-C establishing a peer-to-peer relationship with FEDERICA-D).

Similarly, D and E collaborate via the *peer-to-peer* pattern (e.g. FEDERICA-D establishes a peer-to-peer relationship with PlanetLab-E). This way, PlanetLab users will be able to use FEDERICA's networking resources (dedicated Logical Routers connected with isolated links having a guaranteed bandwidth), while FEDERICA users will be able to use a number of Planetlab resources, i.e. VMs connected over the public Internet.

Finally, E and F platforms collaborate in a peer-to-peer way. For example, E and F could be the PlanetLab and VINI platforms, actually collaborating via the peer-to-peer Slice Facility Architecture (SFA) [2]. More details on SFA will be given in section VI of the paper.

The implementation of mechanisms to establish the relationships between heterogeneous platforms in non-trivial task due to the different resources, services, APIs and management tools of each virtualization infrastructure - member of the federation. In the following section, we will describe implementation issues, using our proposed NDL-extensions as the common Data Model, coupled with resource provisioning considerations.

V. Implementation Issues

A. Extension to the SMC Architecture for Resource Provisioning

Efficient sharing of virtualized infrastructures requires techniques for solving the *Virtual Network Embedding - VNE* problem [19]. VNE provides a mapping of user requests to specific substrate nodes and links . To use VNE algorithms within the SMC Architecture, the latter is extended according to Figure 5. VNE functionality is provided at real time via the *Intelligent Resource Mapping Engine*. This requires input through observations of the states of the overall available substrate resources e.g. the utilization of substrate resources. The engine communicates with the *Monitoring and Context Service* components of the SMC through the *Event Bus*. When needed, it triggers policy rules within the *Policy Service*.

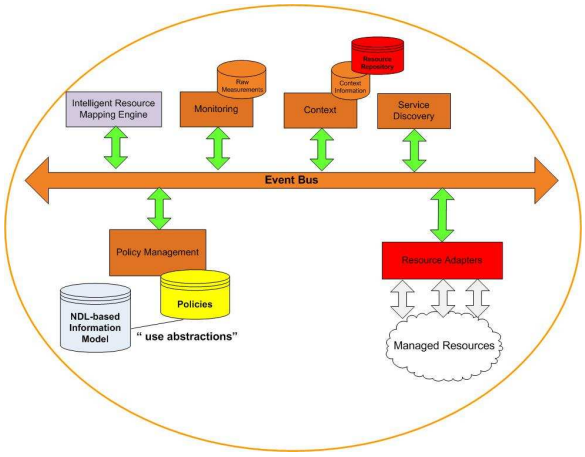

Figure 5. Intelligent Resource Mapping Engine as an Extension to the SMC Architecture

Most of the solutions reported in the literature restrict the VNE problem space by relaxing constraints to reduce its NP-hard complexity. For example [17]takes into consideration only bandwidth constraints. Several researchers (e.g. [17], [20]) handle VNE requests offline. Others decompose it into a node assignment and a link assignment problem. Assigning virtual nodes to substrate nodes is dealt with greedy heuristics. Similarly, link assignment is accomplished either via shortest path algorithms for non-bifurcated flows or multi-commodity flow algorithms in cases of path-splitting [18]. The *Intelligent Resource Mapping Engine* may consider various alternatives for solving the VNE problem. For example, requests concerning individual resources - *slivers*- may lead to the adoption of a greedy node-mapping algorithm, whereas user requests for baskets of resources - *slices*- require solving the full VNE problem via appropriate heuristic algorithms.

B. Policy Management of Federated Virtualized Infrastructures using NDL Extensions

Using the abstractions of our NDL-based data model, we are able to define and deploy domain-independent management policies for users' authorization, resource allocation, reservation and scheduling of resources and services within a federation. Management policies are defined using the Ponder2 policy language, which offers a rich set of expressions for defining both authorization and obligation policies. The latter are Event Condition Action (ECA) rules that can be used for resource and service provisioning operations within a federation. Note here that "low-level" policy actions must be implemented by *resource adapter components*, using diverging interfaces or protocols that various infrastructures implement to provide management and control functionality

of their own resources and services. They could range from CLI-based to custom APIs. This is depicted in Figure 6 below, whereby each virtualized infrastructure may implement its own protocols to communicate with its customized software or hardware substrate.

As defined in the SMC architecture [16], a *mission* defines the requirements of one SMC for interacting with another. A mission is a group of policies which defines the duties of the remote SMC as a set of obligation policies it must enforce. Obligation policies are written according to the *mission interfaces* for each SMC. Mission interfaces specify *Events, Notifications,Local actions* and *Remote actions*.

Figure 6 shows a complex federation structure, employing three (3) virtualized infrastructures, *A, B, D*. Their associated *SMCs: SMC A, SMC B and SMC D* are illustrated in this figure to represent each virtualized infrastructure's policy-based management system. Note that *C* is not a "real" virtualized infrastructure, but a federation of *A* and *B* virtualized infrastructures. This federation is provided since SMC *C* is deployed as the aggregation of SMCs *A* and *B*, using the *aggregation* structural pattern.

Deployed following the aggregation structural pattern, we can see that in the *A,B* federation, the top-level SMC, *C*, acts as a manager for both SMCs *A* and *B*. Thus, the duties of the subordinates SMCs' will be defined as two separate missions, one mission specifying the duties of the SMC *A* in respect to SMC *C* and another mission specifying the duties of SMC *B* in respect to SMC *C*.

As an example, a mission *MissionForA* specifies the duties of the SMC *A* in respect to SMC *C* The mission is comprised of three ECA rules (Policies) and its specification is parameterized by the SMCs' interfaces C_IF and A_IF as shown below:

mission MissionForA(C_IF, A_IF) **do**
1. **on** A_IF.mloaded() **do**
   C_IF.storeSlices(A_IF.getAvaibleSlices())
2. **on** A_IF.sliceRemoved(sliceName) **do**
   C_IF.updateSliceDatabaseA(sliceName, "delete")
3. **on** A_IF.sliceAdded(sliceName) **do**
   C_IF.updateSliceDatabaseA(sliceName, "add")

The SMC *A* generates a *mloaded()* event after the mission has been successfully loaded in it. This triggers Policy 1 within the mission *MissionForA* in order to get all the available slices within the virtualized infrastructure *A* and store this information in a local database within *SMC C*, as shown in Figure 6. This database contains run-time information of the slices running within *A*.This information is needed by *SMC C* is necessary so that it can use its *Intelligent Resource Mapping Engine* for deciding whether and how a slice request for slivers within *A* should be satisfied. Note that Policy 1's action is the remote action *storeSlices* within the *SMC C*. This action is defined within the interface C_IF and is allowed to be called by *SMC D*, through use of authorization policies as we will discuss at the end of this section.

Maintaining the runtime information on running slices within *A* is realized using the obligation Policies 2 and 3. In particular, Policy 2 is triggered within *SMC A* when a slice having the name *sliceName* is deleted from *A*. Policy 2's action is the remote action *updateSliceDatabaseA* within the *SMC C*. This action is defined within the interface *C_IF*. The first parameter of the remote action *updateSliceDatabaseA*is the name of the slice that was deleted, $sliceName$, while the second parameter "$delete$", denotes that the slice should be deleted from the database kept within *SMC C*. Similarly, Policy 3 is triggered to update the database within *SMC C* when a new slice, *sliceName*, is added within *A*.

An analogous mission is defined by the *SMC C* for the virtualized infrastructure *B*, so that *SMC C* maintains the current status of slices within *B*. Having all the slice information from *A* and *B*, resource provisioning for a combined slice request for N *slivers* in *A* and L *virtual links* in *B* (see Figure 4) can be accomplished using a VNE heuristic algorithm. As discussed in sub-section V-A, this algorithm is implemented as a method *Perform_AggregatedSliceAllocation* within the *Intelligent Resource Mapping Engine* component of *SMC C*. It is triggered by the following ECA policy rule:

on CreateFederatedSliceRequest(T) **do**
(N, L) = Perform_AggregatedSliceAllocation(T)

The above ECA rule is triggered when *SMC C* receives a user request *CreateFederatedSliceRequest* for a combined slice with a topology described as object T. Note that, this object belongs to the *Topology* class that we inherit from NDL. User requests could be delivered to *SMC C* through the use of a Web Interface or as calls to *SMC C's* API. In any case, the output of this method reflects the Virtual Network Embedding - VNE of N *Slivers* within *A* and L *Virtual Links* within *B*, in compliance with the NDL extension schema we proposed and discussed in section III.

As we have previously discussed, a decentralized way to structure a federation can be realized using the *peer-to-peer* structural pattern. When this pattern is used, SMCs have an equal role, so there is no manager to agent relationship. Figure 6 presents how a relationship of this type can be established among the *SMC C* and *SMC D*. An example could be that the *SMC D* represents the management system of the GENI ecosystem in the US, while *SMC C* represents the management system of the *aggregation* of the PlanetLab Europe (PLE) [12] and FEDERICA platforms in Europe.

As an example, a mission *MissionForD* could be defined by *SMC C* to specify the duties of *SMC D* in respect to *SMC C*. Another mission, could be defined by *SMC D* for *SMC C*, depending on the agreement between the two management domains. The mission *MissionForD* specification is parameterized by the SMCs C and D interfaces C_IF and D_IF as shown below:

mission MissionForD(C_IF, D_IF) **do**

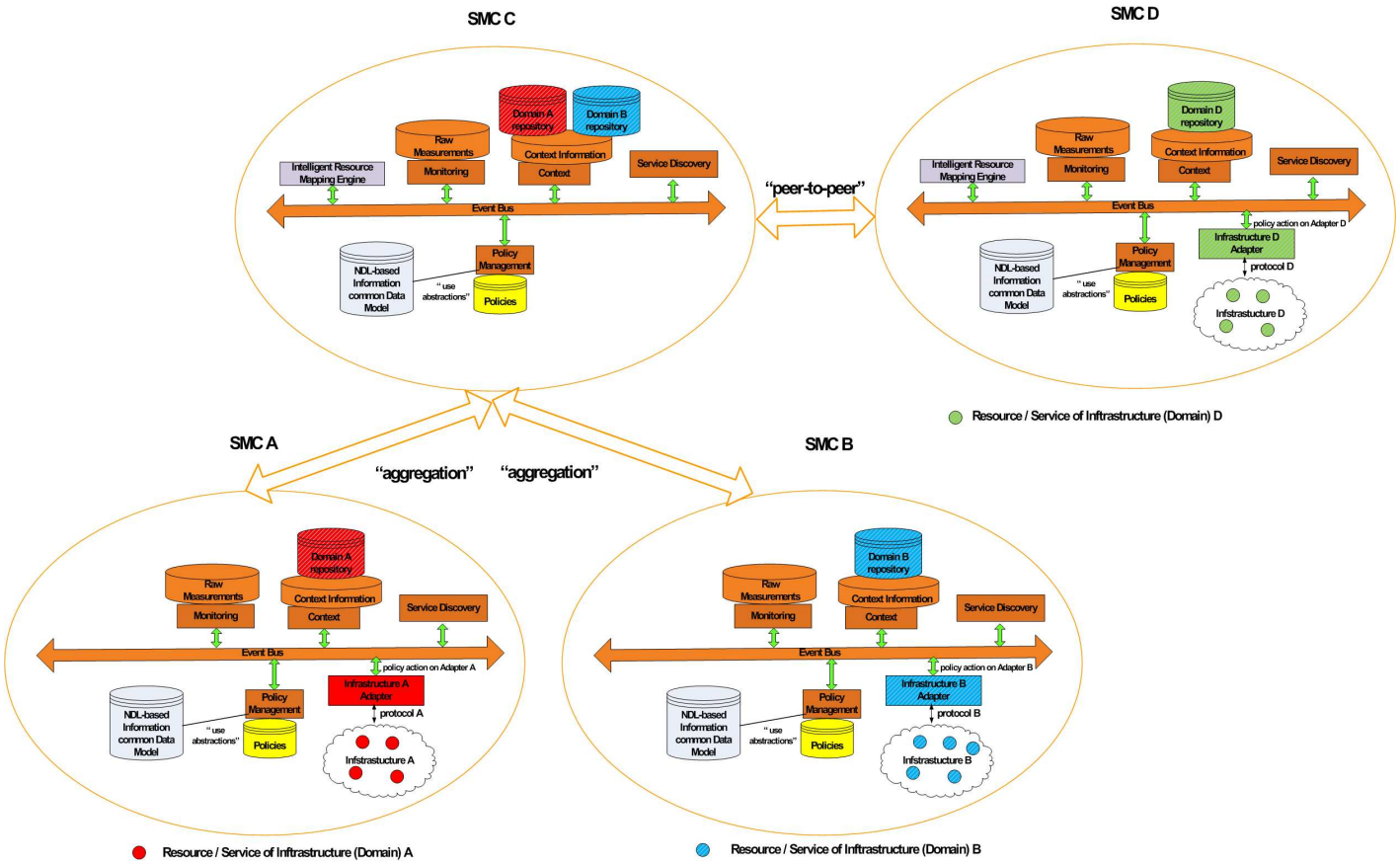

Figure 6. Relationship among SMCs in a Federation

1. **on** D_IF.mloaded() **do**
   C_IF.notify("ok"))
2. **on** D_IF.CreateSliceRequestForSMC_D(Topology T) **do**
   $newSliceWithinDomainD$ = D_IF.PerformSliceAllocation(T)
   if ($newSliceWithinDomainD$!= null)
   then D_IF.CreateSlice(newSliceWithinDomainD);
   C_IF.recvSliceTopology($newSliceWithinDomainD$);

In the above mission, *SMC D* generates a *mloaded()* event after the mission *MissionForD* has been successfully loaded on it. This triggers Policy 1 to notify *SMC C* that the mission is loaded successfully, in a way similar to the mission specification *MissionForA*, described earlier in this section.

Resource allocation for a *Slice* with a topology T in *C* is performed by Policy 2 that uses the *SMC D*'s local action *PerformSliceAllocation*. The event triggering Policy 2 is an external to *SMC C* event, received from *SMC C*. As discussed in [16], events defined within a mission specification can be either local events or events from remote SMCs. In both cases, events are defined within the mission SMC interfaces and are communicated across SMCs with the implementation of a publish-subscribe communication system within Ponder2. Continuing the description of how our example mission operates, if a solution of the VNE problem is found ($newSliceWithinDomainD$) that provides the requested topology T within D, then the *SMC D* creates the requested slice with the local action *CreateSlice* which has as parameter the placement of $newSliceWithinDomainD$, an entity that belongs to the class *Slice* of our proposed common Data Model. Finally, Policy 2 proceeds with the execution of the remote action *recvSliceTopology* within *SMC C*. This action is performed in order that *SMC C* receives the $newSliceWithinDomainD$ entity, i.e. the topology of his/her slice request within the remote management domain *D*. Upon reception of the slice allocated to him/her by domain D, the user will be able to use his/her requested slice within the domain D.

We have not yet discussed how *authorization policies* can be defined and enforced per SMC as a means to allow or deny execution of local methods or actions, invoked remotely by another SMC as a part of their mission. For example, *MissionForD* requires the following *authorization policies* to be enforced:

Authorization policies enforced within *SMC C*:

1. **auth+** D_IF → C_IF .notify
2. **auth+** D_IF → C_IF .recvSliceTopology

Authorisation policies enforced within *SMC D*:

1. **auth+** C_IF → D_IF.loadMission

VI. Related Work

In this section, we will present the most adopted federated management approaches by the FI communities in Europe and the US.

Teagle: Panlab developed the hierarchical Teagle architecture to register and manage resources of federated FI testbeds across Europe within a common repository [3]. According to Teagle, each federation member needs to implement a Panlab Testbed Manager (PTM) component. PTMs have access to a pool of resources which are controlled by resource-specific adapters and allow testbed resources to be exposed as controllable services to Teagle, which provides a tool set for testbed resource management, creation of Virtual Customer Testbeds (VCT) and maintaining a common storage facility (Panlab Repository).

Panlab/Teagle uses the advanced capabilities of the DEN-ng information model [21]. In our work, we do not intend to impose a common data model for all possible FI infrastructures, thus we are currently investigating how DEN-ng can provide us with an advanced information modeling framework that will enable us to capture all necessary abstractions for federating a diverse number of heterogeneous virtualized infrastructures.

Although PanLab uses the advanced capabilities of the DEN-ng information model, the implementation of Teagle has been realized in a centralized manner, employing a common PanLab repository. Our work envisages to be create the algorithms, methods and tools to structure various forms of federations, where by a federation can be seen as a complex graph structure of individual virtualized infrastructures.

PlanetLab Slice-Based Facility Architecture: PlanetLab researchers developed the peer-to-peer Slice-Based Facility Architecture (SFA) [2] to support the federation and interoperability of virtualized platforms of heterogeneous resources (e.g. PlanetLab, EmuLab, VINI of the GENI Initiative). SFA assumes that platforms describe their resources in terms of testbed-specific XML schemas, referred to as RSpecs (Resource Specifications). These are used by the SFA tools to automate sliver/slice management within a federation of platforms.

The two main abstractions defined in SFA are the *component* and the *slice*. A component is a collection of physical and/or logical and/or synthetic resources; a slice is a set of components spanning across the virtualized network infrastructure. These two SFA concepts correspond to the classes *Sliver* and *Slice* of our NDL-based common data model. Thus, custom RSpec schemas can be derived from our RDF-based data model, enabling us to use existing SFA tools.

VII. Conclusions

Section II of this paper presented the Ponder2 Policy framework and the SMC architecture as a powerful engineering approach, which we applied to semantic-aware management of federations of FI virtualized infrastructures. To cater for context-awareness, we proposed in section III a preliminary version of our common information model for virtualized infrastructures, based on NDL, capturing a common set of abstractions of virtualized resources and services. Semantic information can be inferred from monitoring systems and stored in RDF/OWL data models based on our ontologies.

Sections IV and V illustrated how we envisage to use the Ponder2 SMC architectural and engineering patterns for structuring and managing complex forms of federations. To that end, we presented a methodology on how we can use *structural* patterns to model federations as graphs, whose edges denote the type of relationship of the connected vertices. We gave an illustration of a complex graph structure (corresponding to existing FI experimental platforms in the US and Europe) and described examples of *missions* that contain the ECA rules governing inter-working of vertices - SMCs within the federation. Furthermore, we outlined how VNE resource allocation algorithms can be incorporated within the SMC architecture.

In section VI, we presented two major federation approaches a pure hierarchical (Teagle) and a peer-to-peer (SFA). Complementing these approaches, we provide a generalized methodology able to support a mixture of hierarchical and peer-to-peer relations, as elaborated in sections IV and V. Indeed, our approach can be viewed as a generalization of the recent work for inter-working among Teagle and SFA, as reported in [27].

VIII. Future Work

The work reported in this paper opens many opportunities for further research in order to devise algorithms and implement them as components within the SMC architecture. Such extensions will contribute to the Ponder2 Policy framework and may suggest a policy-driven management system tailored to federations of heterogeneous virtualized infrastructures. Within this context, we may specify and evaluate heuristic algorithms for the VNE problem in a federated environment. These will be implemented as methods within the *Intelligent Resource Mapping Engine* component within the SMC architecture.

A second part of our work is to further extend our information model, and the related data model, to include all elements need for federation of virtualized infrastructures.

Finally, it is within our research plans to investigate the problem of how scalable and fault-tolerant resource discovery can be achieved within a federation comprising of a large number of virtualized platforms. Hierarchical clustering schemes could be evaluated, see [29], while

novel schemes could be introduced extending distributed database techniques suggested for peer-to-peer semantic overlay networks [30], [31]. Such approaches could resolve queries on resources described with multiple attributes with a logarithmic complexity on the number of platforms within a large-scale Future Internet federation.

IX. Acknowledgements

This work was partially supported by the European Commission, 7^{th} Framework Programme for Research and Technological Development, Capacities, Grant No. 213107 - FEDERICA and the Future Internet Research & Experimentation (FIRE), Grant No. 257867 - NOVI.

References

[1] NOVI FP7 STREP Project. http://www.fp7-novi.eu

[2] Slice Federation Architecture, v2.0. Available from http://groups.geni.net/geni/attachment/wiki/SliceFedArch

[3] Teagle, http://www.fire-teagle.org

[4] Global Environment for Network Innovations (GENI). http://www.geni.net

[5] FIRE - Future Internet Research & Experimentation. http://cordis.europa.eu/fp7/ict/fire

[6] PlanetLab - An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org

[7] VINI: A virtual network infrastructure, http://www.vini-veritas.net

[8] Emulab - Network Emulation Testbed, http://www.emulab.net

[9] Panlab - Pan-European Laboratory , http://www.panlab.net

[10] FEDERICA - Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures, http://www.fp7-federica.eu

[11] OneLab, http://www.onelab.eu

[12] PlanetLab Europe, http://www.planet-lab.eu

[13] GENI Clearinghouse. Information available from http://groups.geni.net/geni

[14] Morris Sloman and Emil Lupu. Engineering Policy-Based Ubiquitous Systems. The Computer Journal (2010).

[15] Ponder 2 framework. http://www.ponder2.net

[16] E. Lupu, N. Dulay, M. Sloman, J.Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh, A. Schaeffer-Filho. AMUSE: Autonomic Management of Ubiquitous e-Health Systems. Concurrency and Computation: Practice and Experience, John Wiley and Sons.

[17] Mosharaf Kabir Chowdhury,N. M., Boutaba, R., "Network Virtualization: State of the Art and Research Challenges", IEEE Communications Magazine, vol. 47 (7), pp. 20-26, 2009.

[18] Yu, M., Yi, Y., Rexford, J., and Chiang, M., "Rethinking virtual network embedding: substrate support for path splitting and migration." SIGCOMM Comput. Commun. Vol.38 (2),pp. 17-29, 2008.

[19] Mosharaf Kabir Chowdhury, N. M. , Raihan Rahman, M., Boutaba, R., "Virtual network embedding with coordinated node and link mapping." INFOCOM, 2009.

[20] Lu, J., Turner, J., "Efficient Mapping of Virtual Networks onto a Shared Substrate", Technical Report, Washington University in St. Louis, 2006.

[21] Directory Enabled Networks - next generation, http://autonomic-management.org/denng/index.php

[22] Bavier, A., Feamster, N., Huang, M., Peterson, L., and Rexford, J. In VINI Veritas: Realistic and Controlled Network Experimentation. In Proc. SIGCOMM 2006,Pisa, Italy, Sep 2006.

[23] Network Description Language (NDL) - http://www.science.uva.nl/research/sne/ndl

[24] World Wide Web Consortium (W3C), http://www.w3.org

[25] J. van der Ham, P. Grosso, R. van der Pol, A. Toonk, C. de Laat, "Using the Network Description Language in Optical Networks", IM 2007, Munich, Germany, 2007.

[26] S. Bhatia, M. Motiwala, W. Mhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, J. Rexford. "Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware". In Proc. of ROADS 2008/CoNEXT 2008, Madrid, Spain.

[27] K. Campowsky, T. Magedanz, and S. Wahle. "Resource Management in Large Scale Experimental Facilities: Technical Approach to Federate Panlab and PlanetLab." In Proc. of NOMS 2010, April 2010.

[28] Y. Choi, J. Li, Y. Han, J. Strassner, J. Won-Ki Hong: Towards a Context-Aware Information Model for Provisioning and Managing Virtual Resources and Services. In Proc. of MACE 2010, October 2010.

[29] J. Famaey, S. Latrea, J.Strassner, F. De Turck. A hierarchical approach to autonomic network management. In Proc. of the second IEEE/IFIP International Workshop on Management of the Future Internet (ManFI 2010), held in conjuction with the IFIP/IEEE NOMS 2010.

[30] P. Ganesan, B. Yang, and H. Garcia-Molina. One torus to rule them all: multi-dimensional queries in P2P systems. In Proceedings of the 7th International Workshop on the Web and Databases, NY, USA, 2004.

[31] L. Lymberopoulos, C. Pittaras, M. Grammatikou, S. Papavassiliou and V. Maglaris. PLATON: Peer to Peer Load Adjusting Tree Overlay Networks. Paper submitted to Peer-to-Peer Networking and Applications (PPNA) Journal, Springer.

[32] Network Mark-up Language Working Group (NML-WG). Information available at http://www.ogf.org