A Resource Description Language with Vagueness Support for Multi-Provider Cloud Networks

Gregor Schaffrath[1], Stefan Schmid[1], Ishan Vaishnavi[2], Ashiq Khan[2], Anja Feldmann[1]
[1] T-Labs & TU Berlin, Germany [2] NTT DoCoMo Eurolabs, Germany

***Abstract*—The concept of CloudNets, virtual networks connecting cloud resources, has recently attracted much interest from both academic as well as business sides. CloudNets can realize the vision of affordable customized infrastructures. In particular, such networks are expected to be offered even in federated environments with multiple providers. Inter-provider communication about requirements or provisioning of truly customized virtual environments however require a powerful flexible resource description language (RDL). While extensibility and expressiveness seem to be natural requirements for such a language, we identify another less intuitive requirement affecting all actors (or stakeholders) in their economic benefits: the possibility to omit arbitrary specification details and to remain *vague* while at the same time describing real world scenarios. Not only may a description language ignoring this constraint easily become too bulky to use, it is also likely to force players to focus on details they are not interested in or lack the knowledge to map their actual requirements to. This paper identifies detailed requirements for an RDL to allow for topology and requirement communication in business scenarios. Furthermore, we present the FleRD flexible resource description language for multi-provider virtual network architectures. FleRD is fully incorporated in our own CloudNet prototype architecture.**

I. Introduction

Virtualization has become an important paradigm in the development of today's Internet. Decoupling service applications and virtual components from the constraints of the underlying (substrate) physical infrastructures, it facilitates efficient use of the given resources and eases the provisioning and maintenance of a better service quality. The convergence of node and link virtualization technologies opens many new possibilities. Many research efforts (e.g., in the context of testbed virtualization projects [7], [8]) have been started to build isolated virtual service networks over the same physical infrastructure. In the future, customers will be able to quickly request and use arbitrarily specified CloudNets [6], virtual networks connecting cloud resources. These networks are mapped to (and hosted by) the shared physical substrate resources.

In order to surpass the provisioning of simple virtual network topologies and to leverage the potential of truly customized networks, efforts to find flexible ways of describing CloudNets and their properties are underway (e.g., [2], [18]). Properties range from service level over geographical down to embedding constraints (see e.g., [4]) and may possibly even include measurement results. Some of the description languages take ontological approaches in the context of automated reasoning. In contrast, we consider a business scenario in which the language is used for inter-provider communication.

While such a language should facilitate easy translation to ontologies used for reasoning, we advocate only generic objects for communication purposes. Semantic value carried by ontological objects is of local interest to selected entities and components, whereas the syntactic overhead is shared by all. We assume that actors will not only compete on a tariff basis, but tussle about service level abstractions. While one provider may provide bare resources, another one may, e.g., offer managed service infrastructures as part of the product. Both may offer the same network elements, but the second may allow for arbitrary unconventional resource or property descriptions (e.g., bandwidth on hosts, CPU cycles on routers, number of clients on 'Network' abstractions). Frequent syntactic changes of the language or the handling of mutually incompatible description models of competing providers are likely introduce complexity in the operation of, e.g., brokers or relays. A language may allow for these tussles and compensate by relaxing its typification until any property annotation is both possible and optional. This scenario reduces the semantic value of ontological derivative classes to simple 'type' descriptors and in consequence limits the usefulness of the connected syntactic overhead.

A less intuitive requirement is introduced by the insight that it is economically not viable for a single infrastructure owner to have a truly global footprint and presence in every location. Federated provider environments and reselling are likely to form a part in CloudNet scenarios, such as presented in [17]. We assume resellers and providers to lease infrastructure and resources from each other. This introduces more abstraction layers and shows that a plethora of different players will partake in CloudNet provisioning and use.

We expect these players to follow independent goals and be concerned about different aspects of the virtual network and its description. In order to keep a language useful for communication purposes (e.g., for CloudNet requests), it becomes important to allow for the *omission* or at least ignorance of details. Hence another requirement is the possibility to *leave arbitrary details un- or under-specified.*

As an example, consider the following motivating scenario: one customer requesting a CloudNet has specific requirements on host resources (e.g., processing, memory, disk storage) and their geographic placement. She has a basic idea about which hosts need to communicate with which other hosts using specific communication channels. However, she is not interested in how the communication channels are established or their exact details. Another customer may care for complex connectivity constraints but not for the required communica-

tion components. He may wish to directly interconnect uplinks with asymmetric bandwidth with a core link forming a shared medium without explicitly placing switches or adapters in-between. The first customer may even leave host details, such as architecture and word-size, open, but wish for redundant machines to not be co-located on the same underlying (physical) infrastructure components. Both resellers and providers may want to consider the omitted details as wildcards and fill in the gaps in arbitrary parts of the description.

Under-specifying infrastructure details the customers hold no interest in, entails several advantages. All unspecified details form degrees of freedom for optimizations on the provider or reseller side, and may even allow for intra- and inter-provider migration [1]. Requirements limit the choice of substrate components, geographic locations, and possibly even virtualization mechanisms providing different virtual component types. They hence correlate with economic benefits of the providers and, thus, prices for the customers. Furthermore, translation of possibly abstract requirements into technical requirements should happen where competence to do so resides. Considering a service provider renting infrastructure, it is conceivable that the service provider does not know what technical requirements need to be specified in order to guarantee a certain service in a time interval, e.g., what CPU flags may be beneficial. Last but not least, enforcing the specification of irrelevant details may hamper the acceptance of a language and may be perceived as a nuisance.

Following the insight that a usable language needs to be flexible not only in terms of extensibility but also in terms of vagueness, this paper presents FleRD, a ***Fle**xible **R**esource **D**escription language* for virtual networks in multi-provider scenarios. We incorporated FleRD into our virtual network prototype based on the architecture described in [17].

A. Paper Organization

The remainder of this paper is structured as follows. First we identify challenges and requirements for a CloudNet description language in Section II. Based on the compiled wish list, we present the FleRD design in Section III, and subsequently discuss temporal extensions (in Section IV). In order to give an example for the proposed approach, a concrete use case is considered in Section V. We review related work in Section VI and conclude in Section VII.

II. Challenges

A request for a CloudNet may contain specification of elements, placement constraints, connected resources, their topological arrangement, as well as arbitrary other properties (e.g., OS selection), or non-topological requirements referring to more than a single element (e.g., requirements on binary compatibility). We expect the set of possible component types and descriptive details to be ever growing and hard to enumerate at any point in time. Especially the property descriptions may occur on every level of abstraction, depending on the focus of formulating entities. Nonetheless, they should be describable and—as motivated by the multi-provider considerations—optional.

We therefore identify the following challenges and goals.
(1) **Realism and generality:** A description language must be able to describe real world topologies. For example, it is not sufficient to model the network as a graph, where each link has exactly two endpoints. Shared media and shared resources form part of real topologies on every level of abstraction. At the same time, some optimization problems in the substrate (e.g., link placement) are of graph theoretic nature. It would be convenient if the topological description would nonetheless be graph like to facilitate the application of graph algorithmic mapping approaches.
(2) **Extensibility:** The language must be extensible with arbitrary network elements and properties to be future proof. Implementation changes and dictionary updates due to language extensions should be limited to components handling them rather than requiring every component to know them.
(3) **Grouping and aggregation:** The language must allow to correlate properties of multiple network components (e.g., placement or binary compatibility). Moreover, it must be possible (e.g., for resellers) to aggregate resources from different network elements.
(4) **Omission, granularity, and policies:** It must allow for omission of all details irrelevant to the describing entity. Moreover, white or black listing of properties should be supported to allow for different degrees of specifity. The missing details may be filled out during the mapping process, as they are determined. Omission must be possible both for components and their associated properties.
(5) **Mapping:** A virtual component's resources may be split up upon embedding. They may be hosted on different types of substrate resources (e.g., disks onto RAM) or shared resources (e.g., NFS volumes or bandwidth for tunnel connections). In all these cases, mappings should be expressible and unambiguous. They need to clearly identify the substrate components and resources hosting virtual ones.
(6) **Layering:** The mapping may happen in multiple iterations (e.g., to providers, to sites of a provider, on the actual substrate). While we may assume that mappings and embeddings are usually not communicated to business partners, we should allow providers to substructure and gather management information on each mapping step centrally. Moreover, the language should be generic in the sense that it not only describes one virtual and one physical network but also allows for virtualized substrates (e.g., a company providing virtual links via a leased low-latency virtual network). A resource description language used for communication therefore should support the modeling of multilayered mappings.

III. FleRD

We unfold FleRD in three steps. First, we derive design principles from the identified challenges and introduce the basic descriptive objects. Second, we fill in detailed descriptions of individual object fields. Third, we outline the property attribute space we use in the context of our prototype.

A. Approach

In the following, we describe the design approach chosen for FleRD and motivate it referencing to the different requirements

and goals identified in Section II. Throughout the section, we will reference to descriptive objects of FleRD shown in Figure 1.

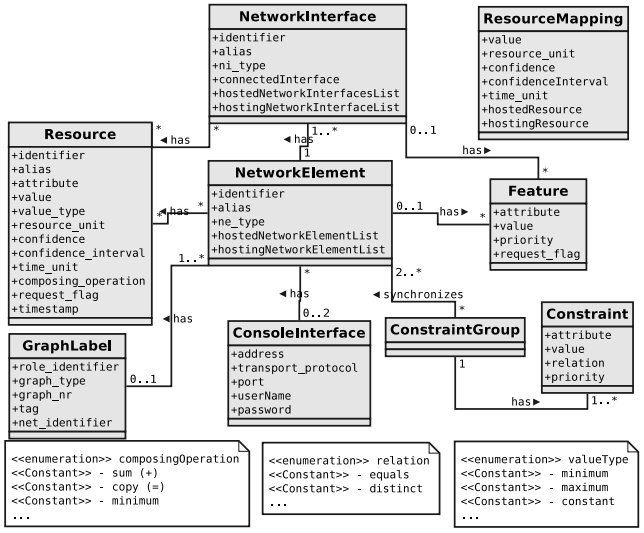

Fig. 1. The FleRD resource description language.

As mentioned and motivated in Section I, we advocate the use of generic description elements in FleRD's context. As a consequence, our description model is centered around basic *NetworkElement* (NE) objects interconnected via *NetworkInterfaces* (NI) objects.

Keeping these objects generic has the side effect that descriptions of resource aggregations, or non-standard entities (e.g., clusters or providers) is trivially supported. They may be modeled as *NetworkElements* of an appropriate type and included as topological elements. This may be used, e.g., to describe mappings in the context of a reseller.

NE properties are described as a set of *attribute-value pair* objects labeled as *Resource* and *Features*. The meaning of resources here is canonic and they may be shared amongst NEs. Features represent any type of property that is not a Resource (i.e., cannot be described in an amount of units; e.g., CPU flags, wordsizes, supported virtualization mechanisms, geographic locations). Associated *Feature* sets are interpreted as a logic clause: *Features* form predicates and sets of *Features* withcorresponding attributes alternatives (disjunctions) within the clause. Features can also be used to explicitly state mapping choices (white listing) or used together with a corresponding attribute to state forbidden mappings (black listing).

While at first glance, the description of shared media (i.e., multihomed) links seems contradictory to the idea of describing topologies as graphs, the concepts can be combined easily: NEs may represent both nodes and links interconnected by *NetworkInterfaces*. Interpreting NEs as vertices and interface connections as edges yields the desired result. Moreover, it becomes possible to interconnect any NE type with any other NE type in the context of the language and hence omit components practically required in embeddings.

While some requesting entity may not care about specific

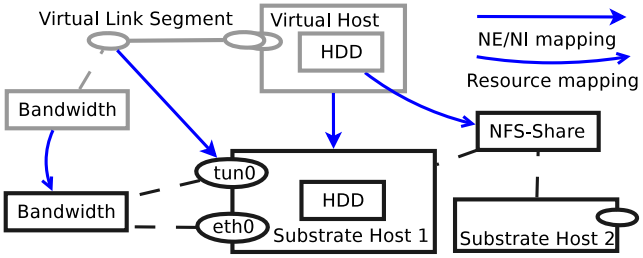

Fig. 2. NE/NI and resource mapping.

property values, it may still want to state correlation constraints: For example, redundant components may not be co-located (geographical or substrate position features), they need to be binary compatible (in architecture, word size, OS), or be diverse (feature value selections should ensure entropy). Also formulations like 'At least one of the machines should be in Washington' may be relevant. FleRD allows for such formulations by means of *ConstraintGroup* objects binding a set of NEs to *Constraint* objects formulating the requirement in terms of feature or resource attributes, optional values and defined correlation functions. Again, *Constraint* sets may be considered predicates in a logic clause to be solved during the mapping steps.

It is technically feasible—and eventually may become economically viable—to introduce multiple layers of virtualization. Conceptually, the substrate does not differ from the CloudNet, as this is the main purpose of the abstraction of virtualization. It therefore seems natural to describe both in the same language. Allowing mappings from one topology to another of the same type allows for recursion and iterative layering, for example in the scenario where an infrastructure is sub-structured. In FleRD the *GraphLabel* object allows for identification of the different descriptions.

Mapping in FleRD follows a dual approach: NE and NI entities may be mapped directly onto other NE and NI entities, and *ResourceMapping* objects convey mapping relations between *Resources*. The rationale for this becomes clear when considering potential ambiguities in Figure 3: *Substrate Host 1* features both local disk space and a NFS volume to accommodate for the virtual HDD resource. Resource mapping is hence required to specify the volume used for hosting. Mapping only the virtual bandwidth resource to the substrate one on the other hand does not allow the identification of the hosting substrate interface.

B. Object Fields

This section provides an explanation to fields of the above introduced description classes. *Resource* objects allow for different type values (describing, e.g., minima, maxima, etc.) in the description of provisioning boundaries. We also include the possibility to specify *units* for the resources as well as *identifiers* (as aliases). Heuristic specifications and mappings are possible, specifying confidence levels and intervals in *Resource* and *ResourceMapping* objects. The *composing_operation* field may specify a function by which mapped (partial) resources aggregate to a given value (e.g., whether

mapped bandwidths correspond to sums or to the minimum of provided partial resources). FleRD offers basic support to convey measurement results as *Resource* objects using the time stamp field. However, we do not recommend to use it for complex measurements for reasons outlined in Section IV.

Formulation of intermediate mappings may be used to express complex resource aggregations (e.g., splitting a link into parallel links and successively mapping links to paths)[1]. *GraphLabels* objects identify each layer description. In our prototype, we differentiate between virtual (overlay OL) network, substrate (underlay UL) graph and interconnecting (mapping layer ML) *graph_type*s. *graph_nr* and a *tag* may annotate a specific sublayer, if intermediate mappings are formulated. The graphs are marked by the identifier of the owner (*role identifier*) and a *net_identifier*. It is important to note that the graph type is relative to the role owning it: an OL graph on behalf of an infrastructure provider may correspond to a partial ML graph of a reseller mapping parts of a CloudNet to different CloudNet providers.

NEs and interfaces feature an *alias* field to allow for local aliasing in different scopes while keeping a common identifier to which all parties can relate, unless otherwise wished and supported by suitable mechanisms. In order to allow for more granular indication of preferences in terms of unfulfillable requests, or to group options together, *Features* and *Constraints* feature *priority* indication fields. To differentiate requests from assignments, if both are to be annotated at the same time, the *request_flag* is used in *Resources* and *Features*.

The optional *ConsoleInterface* objects convey information about customer and provider side interfaces to connect the NEs console. Our prototype uses canonic IP addresses, transport protocol, and port information in combination with username password credentials. This however is just an example.

C. Property Attribute Space

Resource and feature attributes may be structured along ontological concepts (e.g., `/class/subclass/.../fieldname`) to facilitate conversion for automated reasoning. They may be standardized or follow a canonically structured hierarchy analogous to e.g., the Simple Network Management Protocol's (SNMP) MIP structure [13]. Compound attributes specifying sets of properties in one *Feature* object (e.g., with attribute: 'host_config' and value: 'std_offer1') allow to limit the number of needed descriptive objects and may depend on standardization or multilateral agreements.

However, rather than developing a definite attribute scheme, we opted for a canonically extensible attribute structure in the context of our prototype.

In our case, increasing depth corresponds to increasing specificity. Feature attributes therefore usually consist of a part identifying the *NetworkElement* or *NetworkInterface* type, and a second part designating the described property (e.g., `/node/host/.../wordsize`). The resource attribute hierarchy reflects a classification of Resources (e.g. `/link/symmetric/bandwidth`). In order to allow for arbitrary levels of specificity, every level on the hierarchy contains a `.../generic/` subtree, featuring the union of all properties (aforementioned second part) of the respective other subtrees. In order to avoid issues with semantically equal but differently labeled leafs (yielding redundancy in the generic subtrees, possibly inferring unwanted mapping restrictions), we recommend standardization of the leafs and reserved subtrees for private extensions.

[1]This may be used, e.g., to cache steps in mapping decisions, and to save on calculation time when specific parts are changed or migrated

IV. Support for Operational Dynamics

FleRD facilitates the description of requests or mapping states at any point in time. However, as flexibility is one of the main potential advantages of virtualization, it is likely that embedding requirements change over time—either according to a schedule or in reaction to measurement results. This raises the question of whether time annotations are required in the resource description language, and how to incorporate measurements in the description.

We believe that while it may be tempting to include time or measurement information in the description language itself, it may overburden the language. In the case of measurements, a separate dedicated language may be used. This becomes apparent when considering distributed embeddings of a virtual NE: one 'measurement' may include a series of partial measurements or measurement series at different embedding locations. A single relevant parameter may depend on many different basic parameters in a complex manner. The measurement details may need to be conveyed in arbitrary formats and order. They may or may not correlate with individual NEs, network interfaces, *Resources* or *ResourceMappings*. To integrate it in FleRD we suggest a recursively usable generic container for serialized objects which may be associated to every of the above mentioned objects.

As an unobtrusive solution for a basic measurement support, we included the optional time stamp field in the *Resource* object. This constitutes a basic means to convey actual embedding situations in comparison to requested and promised resources. Our prototype architecture [17] assumes requirement driven communication between CloudNet customers and providers based on business concerns. Customers will not receive information about or access to substrate components by default. The requirements are either directly based on measurables in the provider domain, or relate to parameters the provider is able to correlate with the measurables (if more than simple resources are offered). Measurements outside one's own domain (i.e., breaking the abstraction of virtualization) is thus considered an optimization involving proprietary mechanisms between selected entities. We therefore consider it out of scope for generic languages.

The inclusion of time constraints may not only warrant extensions to all objects but imply additional side effects: Property changes or temporary removal of components may result in nontrivial topological changes, which have to be expressed as well. On the other side, description of multiple full graphs annotated by time constraints may yield the same expressiveness at the cost of redundant specification parts. In principle, it would be possible to use the FleRD *Feature* object to specify basic time constraints on NE granularity, and to

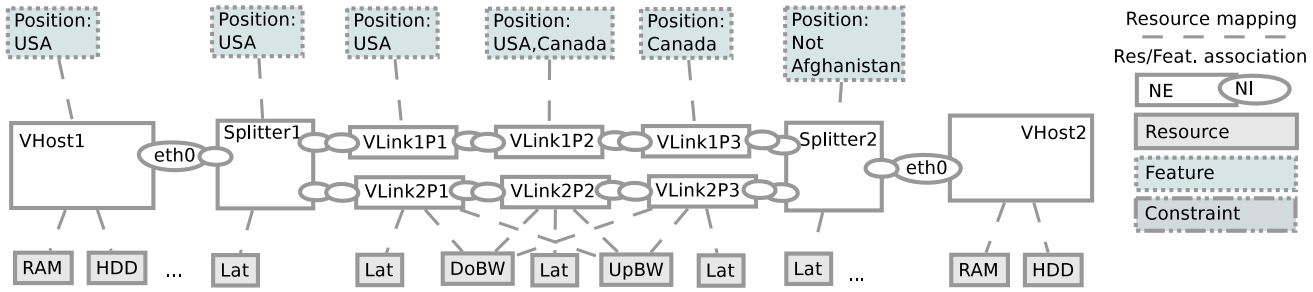

Fig. 3. Mapping layer.

identify coherent topology versions, e.g., by corresponding tags but we do not recommend this use. Given that an annotation of time constraints on a graph level would be trivially feasible by integration into negotiation interfaces, we advocate such a solution.

V. A Webservice Use Case

In order to give an example, in this section, we attend to a concrete use case. For the full details of the use case presented in this section, we refer to reader to the FleRD XML schema at our project website, see http://www.net.t-labs.tu-berlin.de/˜stefan/virtu.shtml.

We consider a CloudNet request (e.g., issued by a service provider) consisting of two virtual hosts *VHost1* and *VHost2* that are connected by two interfaces. For example, this CloudNet may describe a *webservice*, offering some service (of type A), and which is distributed over two processes. The CloudNet specifies that *VHost1* can either be located in the U.S. or in Canada (*white listing*), while the location of *VHost2* is completely unspecified (i.e., *vague*). Moreover, the interfaces connecting the two virtual hosts forbid a connection via Afghanistan (*black listing*), and specify that the connection must be redundant in the sense that it is realized as two disjoint paths.

In the following, we will describe the different stages of mapping a CloudNet request to a physical underlay network. We divide the process into different layers, two request layers (*Overlay0* and *Overlay1*), a mapping layer to, e.g., annotate configuration details, e.g., VLAN identifiers, or to keep mapping state, and a substrate layer (in our case split into an *Underlay0* and an *Underlay1*). We assume the perspective of a single business role, but the requests could also be communicated between different business roles. Note also that several layers could be merged, but we purposefully keep them separate to illustrate the concepts.

Figure 4 illustrates the Network Elements, Resources, Features and Constraints of the CloudNet request (the so-called *Overlay0*). To give a concrete example, in an XML-based version of our language, the white listing is described as follows:

```
<features>
<! − − X: whitelist
− this node may be hosted in the US (preferred) or Canada − − >
  <Feature>
    <attribute>/position/continent/country</attribute>
    <value>/NA/USA</value>
    <priority>1</priority>
    <request_flag>true</request_flag>
  </Feature>
  <Feature>
    <attribute>/position/continent/country</attribute>
    <value>/NA/Canada</value>
    <priority>2</priority>
    <request_flag>true</request_flag>
  </Feature>
</features>
```

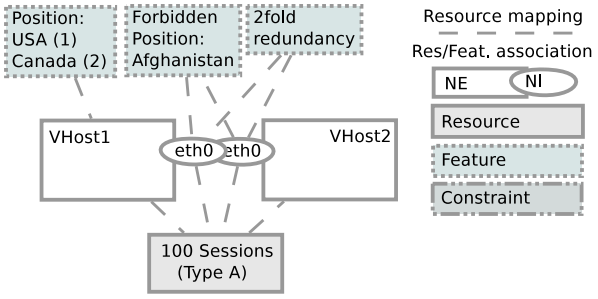

Fig. 4. Overlay0 (OL0).

The original CloudNet request is subsequently expanded and complemented with concrete Resource annotations (e.g., by a virtual network provider): the virtual hosts have RAM and HDD resources and the virtual links have requirements for upstream and downstream bandwidths as well as for the tolerable latency. In order to realize the requested redundant link, two multiplexing components are added to split the link into two disjoint virtual paths. A Constraint is used to ensure that the virtual links are mapped along disjoint paths, i.e., along resources with physically distinct positions. Figure 5 summarizes the resulting request (of *Overlay1*).

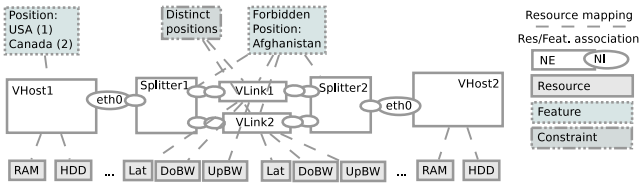

Fig. 5. Overlay1 (OL1).

Both *Overlay0* and *Overlay1* represent requests, i.e., the *request_flag* is set. The remaining layers do not set this flag in our example. A mapping layer constitutes the “bridge”

between overlay request and substrate. It is optional but can simplify the mapping, and can contain annotations of the mapping or also store state (e.g., on which elements have been communicated to which other business roles). In Figure 3 an example is given, where the virtual links are expanded into three virtual hops, one in the U.S., one in Canada, and one between the two countries.

Finally, FleRD is used to describe the underlay. In our case, it is split into two parts, an *Underlay0* and an *Underlay1*. (Note that in our terminology, Underlay1 is *above* Underlay0.) One reason to have two underlays is to keep both a representation of the logical structure of the substrate, as it is used to map CloudNets, and a representation of its actual physical setup. E.g., Underlay1 still contains a (VPN) *tunnel* and a shared NFS. Figure 6 shows Underlay1: In the substrate network there are three hosts *SHost1*, *SHost2*, *SHost3*. In our example, *VHost1* is mapped to *SHost3*, *Splitter1* is mapped to *SHost1*, and *VHost2* and *Splitter2* are mapped to *VHost2*. We assume that the upper virtual link is realized via three tunnel segments, and is routed via two external physical providers, *Provider 2* and *Provider 3*. The second virtual link is realized inside the given provider as a substrate link. Also note that the network file system NFS is still logically attached to two substrate nodes.

In Underlay0, finally, the physical situation for the considered provider is described. We see that of course the NFS is hosted only by one host, namely *SHost3*, and an additional substrate link is used to provide *SHost1* access to the NFS. Moreover, the physical counterparts of the tunnel are not visible by the considered provider.

VI. Related Work

Any shared testbed and federated prototype architecture requires a resource description languages to describe and communicate network specifications, services as well as embeddings. PlanetLab [3] uses the *Resource Specification* (RSpec) language to communicate network requirements between various actors. RSpec [16] is specified under the *ProtoGENI* project and is a basic resource description language comprising *nodes* and *links*. Nodes may be bound to a certain *location*, have various *interfaces*, have a *relation* with other nodes, and may host certain services. The *Network Description Language* [19] approach is based on the *Resource Description Framework* [15]. The original intent of this effort was to describe physical networks (in particular optical networks) to facilitate inter-domain lambda provisioning [20]. Efforts within the *NOVI* [10] and the *GEYSERS*[2] project are underway to extend the language to include the description of virtual networks. The NDL+OWL [2] language is based on an ontology paradigm which allows for automated reasoning and calculation of substrate configurations. Houidi et al. [9] propose an ontology which is also centered around the concept of a *NetworkElement*, but the authors do not consider the additional aspects covered in this paper, such as vagueness, omission, and non-topological constraints such as binary compatibility or co-location. Moreover, mapping remains restricted to two layers (virtual and physical).

The *Open Cloud Computing Interface* (OCCI) [11], [12], [14] was originally specified by the Open Grid Forum (OGF) as an API for services based on the infrastructure-as-a-service model. It has since undergone many extensions to serve services based on platform-as-a-service and system-as-a-service models. The OCCI specification consists of the *OCCI Core* [14], *OCCI Infrastructure* [11] and the *OCCI HTTP Rendering* [12] standards. The OCCI core specification and the infrastructure specification describe how to model an entity and the associated resources and links respectively, whereas the serialization of these models is demonstrated in the OCCI HTTP Rendering standards.

DEN-ng [18] is based on the DEN information model and focuses on business-driven network management solutions. *VN-SLA* [5] allows the formulation of *Service Level Specifications* (SLS) for virtual networks. Specifically, it allows to specify consequences of failure to meet specified service levels. *VxDL*[3] is a virtual private execution infrastructure description language. VxDL describes computing and management resources while including the time limits for which these resources need to be realized. Any description in VxDL consists of four parts: general description, description of non-network resources, network topology and relevant resources, and finally the time line for which the description is valid.

These languages differ from our approach in several respects. Ontological approaches, such as NDL+OWL allow for reasoning about substrate configurations. FleRD however focusses on communication in scenarios where flexibilities required for business tussles are likely to render the associated syntactical overhead a burden. In contrast to the other languages, FleRD explicitly allows for structured multi-layer mapping descriptions on both network element and resource granularity. While VxDL and VN-SLS allow the modeling of CloudNets, they do not consider the formulation of mappings. DEN-ng allows for mapping, but differentiates between virtual and physical instances, thereby limiting the depth of the description. NOVI mentions the possibility of virtualized substrates, but does not consider resource mappings. Like NDL, it also allows the modeling of a (multi-homed) shared communication channel only as a full mesh of distinct links by limiting links to two endpoints. While RSpec allows for the interconnection of multiple endpoints via one shared link, resources are not modeled as separately referenceable entities. This complicates the specification of resources shared between two or more nodes. Moreover, we are not aware of any literature in the field explicitly addressing specification granularity, allowing for white and black listing of properties, or omission of both resources and components in a federated environment.

VII. Conclusions

We believe that the CloudNet paradigm will change the way the different players in the Internet interact. Furthermore, virtualization may also lead to new roles such as brokers

[2]See http://www.geysers.eu/.

[3]See http://www.ens-lyon.fr/LIP/RESO/Software/vxdl/home.html.

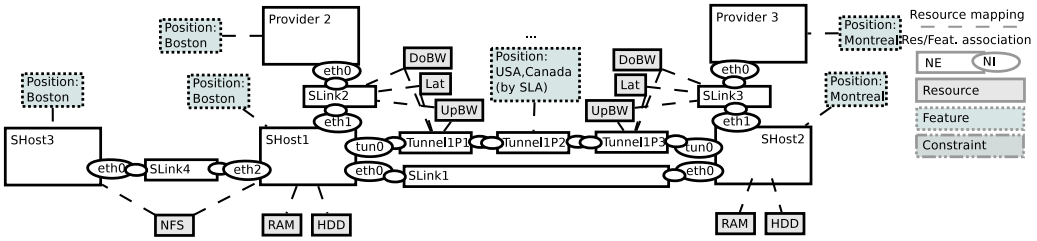

Fig. 6. Underlay1 (UL1).

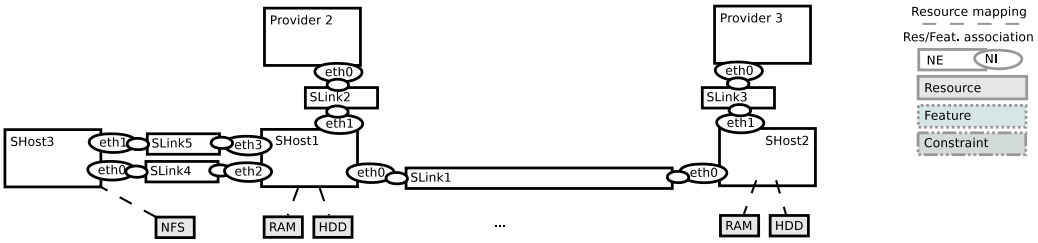

Fig. 7. Underlay0 (UL0).

that aggregate resources from multiple infrastructure providers to offer cross-provider solutions. In particular, we believe that providers will compete not only on a tariff basis, but also on the abstraction (i.e., type) of offered services. This implies frequent non-standard descriptions and emphasizes the need for a flexible and formal resource description language supporting communication of arbitrary requirements. Moreover, we find that specifity in requests comes at a cost in embedding flexibility, yielding an economic incentive to allow for vagueness in both request formulations and mapping assertions. This paper discusses the design of a resource description language as a first class citizen and proposes the FleRD language which meets these requirements. We have also shown some significant advantages of FleRD over previously existing models. Our language is currently utilized in a CloudNet testbed infrastructure deployed across the T-Labs and NTT DoCoMo labs in Berlin and Munich.

ACKNOWLEDGMENTS

The authors would like to thank Johannes Grassler for many discussions and for incorporating FleRD in the prototype.

REFERENCES

[1] D. Arora, M. Bienkowski, A. Feldmann, G. Schaffrath, and S. Schmid. Online strategies for intra and inter provider service migration in virtual networks. In *Proc. Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 2011.

[2] I. Baldine, X. Yufeng, A. Mandal, C. Renci, U. Chase, V. Marupadi, A. Yumerefendi, and D. Irwin. Networked cloud orchestration: A GENI perspective. In *Proc. GC Workshops*, 2010.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM CCR*, 33(3), 2003.

[4] G. Even, M. Medina, G. Schaffrath, and S. Schmid. Competitive and deterministic embeddings of virtual networks. In *Proc. 13th International Conference on Distributed Computing and Networking (ICDCN)*, 2012.

[5] I. Fajjari, M. Ayari, and G. Pujolle. VN-SLA: A virtual network specification schema for virtual network provisioning. In *Proc. 9th International Conference on Networks (ICN)*, pages 337–342, 2010.

[6] A. Feldmann, G. Schaffrath, and S. Schmid. Cloudnets: Combining clouds with networking. *ERCIM News*, (88):5657, 2012.

[7] G-lab. http://www.german-lab.de/.

[8] Global Environment for Network Innovations. http://www.geni.net.

[9] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke. Virtual resource description and clustering for virtual network discovery. In *Proc. ICC Workshops 2009*, pages 1–6, 2009.

[10] L. Lymberopoulos, P. Grosso, C. Papagianni, D. Kalogeras, G. Androulidakis, J. van der Ham, C. de Laat, and V. Maglaris. Managing federations of virtualized infrastructures: A semantic-aware policy based approach. In *Proc. 2011 IFIP/IEEE ISINM*, 2011.

[11] T. Metsch and A. Edmonds. Gfd 184: Open cloud computing interface-infrastructure. *Open Grid Forum Standards*.

[12] T. Metsch and A. Edmonds. Gfd 185: Open cloud computing interface-http rendering. *Open Grid Forum Standards*.

[13] D. Perkins. Understanding SNMP MIBs. In *SynOptics Communication*, 1993.

[14] N. Ralf, A. Edmonds, A. Papaspyrou, and T. Metsch. Gfd 183: Open cloud computing interface-core. *Open Grid Forum Standards*.

[15] Resource description language. http://www.rfc-editor.org/rfc/rfc3870.txt.

[16] Rspec v.2. http://www.protogeni.net/trac/protogeni/wiki/RSpec.

[17] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proc. ACM SIGCOMM VISA*, 2009.

[18] J. Strassner. Den-ng: Achieving business-driven network management. In *Proc. IEEE/IFIP NOMS*, 2002.

[19] J. Van der Ham, F. Dijkstra, F. Travostino, H. Andree, and C. de Laat. Using rdf to describe networks. *Elsevier FGCS*, 22(8), 2006.

[20] J. van der Ham, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat. Using the network description language in optical networks. In *Proc. IFIP/IEEE ISINM*, 2007.