Networked Cloud Orchestration: A GENI Perspective.

Ilia Baldine, Yufeng Xin, Anirban Mandal, Chris Heermann
RENCI, UNC-CH
Jeff Chase, Varun Marupadi, Aydan Yumerefendi
Department of Computer Science, Duke University
David Irwin
University of Massachusetts, Amherst

***Abstract*—This paper describes the experience of developing a system for creation of distributed linked configurations of heterogeneous resources (slices) in GENI. Our work leverages a number of unique architectural solutions (distributed architecture, declarative resource specifications, unique approach to slice instantiation) which is applicable to a wider set of problems related to autonomic co-scheduling and provisioning of heterogeneous networked resources. We discuss the architecture, the resource description mechanisms and some of the algorithms used to enable our system. We conclude with an analysis of a real experiment at allocating resources from multiple providers across a very wide geographic area (spanning Massachusetts, Illinois and North Carolina) to create a single private Layer 2 network connecting virtual machines on the campus of Duke University to a sensor testbed at University of Massachusetts, Amherst.**

I. Introduction

Pervasive virtualization at the edge and in the network core drives the evolution of the IT infrastructure towards a service-oriented model [1]. It permits a move from static arrangements of resources that persist over long periods of time to highly dynamic arrangements that respond to the needs of customers by dynamically provisioning necessary network and edge resources with some notion of optimality. Clouds are one type of virtualized service offered as a unified hosting substrate for diverse applications, using various technologies to virtualize servers and orchestrate their operation. Emerging cloud infrastructure-as-a-service efforts include Eucalyptus, Nimbus, Tashi, OpenCirrus, and IBM's Blue Cloud. Extending cloud hosting into the network is a crucial step to enable on-demand allocation of complete networked IT environments.

Other types of "substrate" - storage and networks, are also developing control approaches that allow them to be subdivided and/or virtualized and are offering those capabilities through various control interfaces. Various existing storage virtualization system, either host-based, device-based or network-based, present the user a logical interface to the storage space and handle the mapping to the physical devices. In network virtualization mechanisms (circuits, wavelengths etc) are exposed via control planes such as MPLS, GMPLS, DOE IDC/OSCARS, NLR's Sherpa, Internet2's DRAGON and the emerging work on standardized network interface through OGF NSI.

The next frontier in this work is enabling the creation of cloud networks: orchestrated arrangements of *heterogeneous* resources (compute, storage, networks, content, scientific instruments etc.), through a single interface. Some work is ongoing in this area [2], [3], [4] primarily centered around connecting specific substrate types, like Grids across dynamic circuit networks. In this paper we present an alternative approach, with a *meta* control architecture designed independently of any a priori substrate assumptions and capable of driving multiple heterogeneous substrates using their native control/management interfaces and creating orchestrated arrangements of these resources *acquired from multiple independent providers*.

This paper reports on a RENCI-Duke collaboration (*http://geni-orca.renci.org*) to build a testbed for the Global Environment for Network Innovation (GENI) Initiative recently launched by the National Science Foundation and managed by BBN. GENI (*http://www.geni.net*) is an ambitious futuristic vision of Infrastructure as a Service as a platform for research in network science and engineering. A key goal of GENI is to enable researchers to experiment with radically different forms of networking by running experimental systems within private isolated *slices* of a shared testbed substrate. A GENI slice gives its owner control over some combination of virtualized substrate resources collected from different providers and assigned to the slice, which may include virtual servers, storage, programmable network elements, networked sensors, mobile/wireless platforms, and other programmable infrastructure components attached to the cloud network. GENI slices are built-to-order for the needs of each experiment.

We focus on progress in building a unified control framework for a prototype GENI facility incorporating RENCI's metro-scale network testbed called BEN (Breakable Experimental Network) [5] with available edge substrates, such as computational resources and various testbeds (primarily sensor and wireless) also complemented by experimental national network backbone (National Lambda Rail, NLR). Each of these resources has distinct attributes and uses different control mechanisms. Our control framework, called ORCA (for Open Resource Control Architecture) unifies these multiple disparate mechanisms under a single architecture and enables *autonomic* control of these heterogeneous resources belonging to different providers. For this reason, we believe our results are applicable to a wider set of problems and audiences beyond GENI that

are related to autonomic co-scheduling and provisioning of heterogeneous networked systems.

We have recently demonstrated a key milestone: on-demand creation of complete end-to-end slices which include resources allocated at multiple sites (RENCI, Duke, and UNC, UMass Amherst). Inside the slices edge resources (compute cloud and sensor testbed) are connected via private IP network, which is configured on top of VLANs. VLANs are dynamically instantiated from multiple independent network providers (BEN, NLR Framenet and campus providers) and *stitched* together to form a single circuit. In the context of GENI, this capability enables a researcher to conduct safe, reproducible experiments with arbitrarily modified network protocol stacks on a private isolated network that meets defined specifications for the experiment. It also has wider implications of demonstrating architectural concepts for a system capable of managing networked heterogeneous resources and creating purpose-built configurations out of those resources.

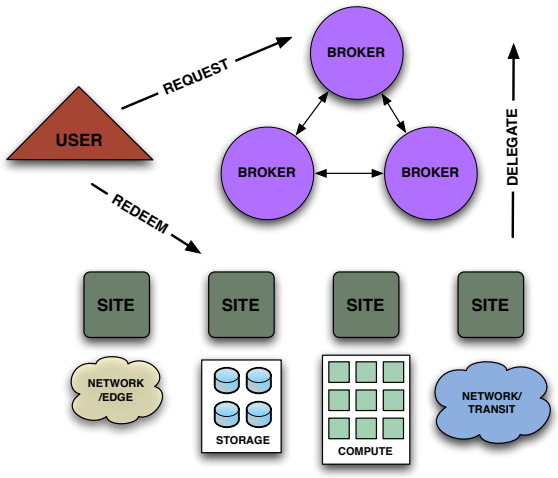

Fig. 1. ORCA architecture.

II. A Control Framework for Heterogeneous Resource Control

Our ultimate goal is to manage the network substrate as a first-class resource that can be co-scheduled and co-allocated along with other resources, to instantiate a complete built-to-order network slice hosting a guest application, service, network experiment, or software environment. The networked cloud hosting substrate can incorporate network resources from multiple transit providers and server hosting or other resources from multiple edge sites (a multi-domain substrate). In the long term we see adding scientific instruments, digital content and datasets, in-slice measurement capabilities and other unique substrate types into the mix.

The two primary challenges we are addressing in our work are that (**a**) the resources can come from different independent providers and (**b**) that resource types are heterogeneous. The providers may choose to contribute some subset of their total resources to a pool, from which configurations can be drawn to create 'slices' for experiments, computations or other distributed activities. The process of creating a slice is fully autonomic and includes computing a feasible configuration based on user request, co-scheduling the resources from multiple providers and configuring allocated resources for the specific task (an example may be installing a specific OS image on bare hardware or into VM).

Our control framework software is based on the Open Resource Control Architecture (ORCA) [6], [7], [8], [9], [10], [11], an extensible platform for dynamic "leasing" of resources in a shared infrastructure. The ORCA platform is in open-source release as a candidate control framework for GENI, and is also a basis for ongoing research on secure cloud computing and autonomic hosting systems. ORCA approaches the described challenges using a combination of architectural solutions.

The basic structure of the framework is shown in Figure 1. Substrate providers are represented by *site* actors, which securely delegate their resources to one or more *brokers*. Brokers are containers for resource allocation policies. User requests for various resources to brokers are satisfied in the form of tickets, which are then presented by the user to different sites in *redeem* operation. Sites then allocate and configure resources described in tickets. Users can also perform additional configuration actions on the issued slices to further customize the substrate to their needs. All messages are cryptographically signed to ensure a verifiable resource delegation path (from sites, to brokers, to users). Any number of actors can be present in the system and they communicate with each other using pre-defined messaging primitives. Various arrangements of brokers are possible, ranging from fully centralized to fully distributed (requiring distributed agreement to support resource co-scheduling).

This distributed architecture answers well to the challenge of distributed independent resource providers. We have identified a number of problems associated with the challenge of resource heterogeneity. The first is making the control framework *understand and reason* about the available substrate, i.e. being able to match user requests based on either specific or vague descriptions of a desired configuration to the available substrate. This includes the ability to easily extend the approach to new resource types as they become available. Second is the problem we generally refer to as 'stitching' - connecting different pieces of substrate from different providers together into a single slice. These are the two problems we are concentrating on and our approaches to them are described in this article.

There are many related questions that can be asked of the control framework, such as: How to protect the security and integrity of each provider's infrastructure, and protect hosting providers from abuse by the hosted guests? How to verify that a slice built to order for a particular guest is in fact behaving as expected? How to ensure isolation of different guest slices hosted on the same substrate? How to provide connectivity across slices when connectivity is desired, and police the flow of traffic? How to efficiently abstract resource representations to make inter-domain path computation scalable? We leave these as a subject of ongoing and future studies.

III. Language and Policies for Heterogeneous Resources

One focus of the project is to advance standards and representations for describing network cloud substrates. The current dominant form of information exchange for resource allocation systems is based on XML and XML schema. However, XML schemas only define the syntax to the resource description and lack a formal way to specify the semantics. Recent developments in Semantic Web technologies enable machine comprehension of resource descriptions and automatic service provisioning (discovery, composition, and interoperability) [12]. Declarative semantic resource descriptions represent an alternative to the XML-based specifications [13]. They express the knowledge about the world by stating relationships between the elements of the world and rely on dictionaries of terms (referred to as ontologies), their attributes and the types of relationships and constraints on them. RDF-based models easily support the merging of descriptions of various substrates based on different dictionaries.

There is a need for a common declarative language that can represent multi-layered physical network substrate, complex requests for network slices, and the virtualized network resources (e.g., linked circuits and VLANs) leased for a slice, i.e., allocated and assigned to a slice. Ideally, we could specify all substrate-specific details declaratively, so that we can incorporate many diverse substrates into a network cloud based on a general-purpose control framework and resource leasing core. Rich declarative resource specifications present a layer of abstraction between service providers and users on one side and the control framework on the other. Critically, the same representation should be used by users requesting resources from brokers to describe the desired configuration, for substrate advertisements by substrate owners, and for slice specification returned by substrate owners to the users.

Declarative representations are difficult in this domain because of the need to express complex relationships among components (e.g., network adjacency), properties and constraints of each network level, and constraints involving multiple levels. Our approach extends the Network Description Language (NDL [14]). NDL representations are documents in RDF (Resource Description Framework), a way of describing sets of objects and their properties and relationships (predicates). NDL is an *ontology*: a set of resource types and relationships (properties or predicates) that make up a vocabulary for describing complex multi-layered networks in RDF. NDL is based on ITU-T G.805 abstraction for connection-oriented networks [15]. An NDL document uses the NDL vocabulary to specify a set of resource elements (in this case links, nodes, interfaces and layer adaptations) and relationships among them, whose meanings are defined by NDL. NDL has been shown to be useful for describing heterogeneous optical network substrates (GLIF) and identifying candidate cross-layer paths through those networks [16], [17].

Figure **??** demonstrates the complexity of the problem. Today's optical network consists of multiple layers (fiber, optical transport, layers 2 and 3) with adaptation functions between layers. Higher layer topologies are embedded in the lower layer topologies, thus creating client-server relationships. For a networked cloud resource management system it is essential to be able to manage these layers jointly to allow connectivity options at layers other than Layer 3 (IP) as well as to allow for optimal management of network resources between the layers. Path-finding in multi-layered environments has been shown to be NP-hard [18].

One contribution of the project is to extend NDL to use a more powerful ontology defined using OWL (Web Ontology Language), an extension we call NDL-OWL. The power of OWL derives from a richer (compared to RDF) vocabulary for defining relationships among the resource types and among the predicates in the ontologies that it describes. In addition to hierarchical classes and predicates, OWL introduces logic-expressive capabilities including class constraints like *disjointness*, *intersection*, *union*, and *complement*, property constraints like *transitive*, *symmetric*, *inversive*, *cardinality*, etc.

An OWL ontology uses these capabilities to define the structure and relationships of predicates and resource types that make up the ontology's vocabulary. Given knowledge of these relationships in an ontology, an *inference engine* can ingest an RDF document based on the ontology, and manipulate it or infer additional properties beyond those that are explicitly represented in the document. For example, in NDL-OWL, the *hasInterface* and *interfaceOf* properties are related in the ontology using the *inverseOf* property axiom in OWL: thus software can infer the property in one direction from a statement that the inverse property holds in the other direction. We use the *Transitive* property axiom in OWL to define interface connectivity and adaptation properties. These features are useful for path finding algorithms. For example, if a sequence of pairs of points are connected, an end-to-end path can be inferred using the transitivity of a *connectedTo* predicate.

The advantage of RDF/OWL approach is easy extensibility. Unlike XML-based approaches, no modifications to the schema are required. The extension is performed by adding new types of objects and relationships into the dictionary, so they can be used to describe new types of resources. We have extended NDL to enable it to describe various compute capabilities. Current NDL-OWL, for example, includes the definition of simple compute resources. It has a class called *UnitServer* which defines a single server, with two floating point parameters *cpuCapacity* and *memoryCapacity* (processor speed and memory size, respectively). Both are restricted to be floating point values:

```
<owl:Class rdf:about="#UnitServer">
    <rdfs:subClassOf rdf:resource="#ClassifiedServer"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#cpuCapacity"/>
            <owl:someValuesFrom rdf:resource="&xsd;float"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#memoryCapacity"/>
            <owl:someValuesFrom rdf:resource="&xsd;float"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Below is a definition of a *BENUnitServer* which is an 'instance' of the *UnitServer* class and is a typical server found in BEN network - a 1.8Ghz single-core with 4GB of RAM. The *topology:hasInterface* stanza from the topology dictionary describes the connectivity of the server to an aggregator switch.

```
<compute:UnitServer rdf:about="#BENUnitServer">
    <compute:cpuCapacity rdf:datatype="&xsd;float">1.8
    </compute:cpuCapacity>
    <compute:memoryCapacity rdf:datatype="&xsd;float">4.0
    </compute:memoryCapacity>
    <topology:hasInterface
    rdf:resource="#UNC/Euca/GigabitEthernet/1/ethernet"/>
</compute:UnitServer>
```

It is worth noting that RDF-XML format for expressing RDF and OWL [13], shown above, is only one of several possible syntaxes in which OWL models can be expressed. NDL-OWL is currently used to request resources from the system and by sites to advertise resources. In the course of the system operation, the NDL-OWL model is loaded from a static file expressed in RDF-XML format that describes the initial state of the resources. This model is queried and updated in-memory to reflect the current state of the network (e.g. connections coming and going). The ultimate goal of this process is to create a representation language that is sufficiently powerful to enable generic resource control modules to reason about substrate resources and the ways that the system might share them, partition them, and combine them. Each resource control action, such as allocating or releasing resources for a slice, affects the disposition of the remaining substrate inventory. To meet our goals, the declarative representation must also capture these substrate-specific constraints on allocation and sharing. These constraints are crucial for the resource control plug-in modules in ORCA, which are responsible for allocating and configuring substrate resources for each slice.

IV. THE STITCHING PROBLEM

One of the major tasks for resource orchestration system such as ORCA is stitching different pieces of virtualized resources from geographically distributed compute, network, and storage substrate into a single connected configuration. These configurations/slices are isolated from one another by means of labels. In networks, various labels are used to isolate and identify logical network segments (e.g. VLAN IDs for Ethernet network, IP subnet for IP network, etc) or physical network segment (frequency channels in optical network and wireless network). In storage systems, logical unit number (LUN) is used to identify a device or a fibre channel storage port in a storage area network. In order to form isolated and networked end-to-end infrastructure on demand, the different pieces of resources from different providers must be stitched together by *negotiating labels between neighboring domains*. In this case by 'domain' we mean substrate owned by a single administrative entity. Edge domains generally represent compute and storage resources, while transit domains provide network connectivity.

VLANs represent the basic unit of connectivity in GENI. VLAN service is also widely available from research backbones and commercial providers. Significant standards work is extending the monitoring and management capabilities of VLANs to allow them to operate reliably in multi-provider setting: the GMPLS based Ethernet Label Switching (GELS) standard in IETF and the PBB-TE (Provider Backbone Bridge Traffic Engineering) to name a few. VLAN tags are the most common type of label we have encountered in this project, however our work easily generalizes to any other type of label mentioned above.

In the course our work we have attempted to make no restrictive assumptions about the behavior of the domains. In our view domains can produce and/or consume labels. Label producing domains generate labels that their neighbors can accept and either convert or pass through. A label producing domain may be a transit domain that creates a bandwidth-guaranteed VLAN between two of its interfaces with a specific VLAN tag. To be used in a slice this VLAN tag needs to be consumed by the neighbor domains and either accepted, translated or passed through (e.g. attaching resources to a VLAN tag in an edge domain, VLAN tag translation or VLAN tunneling in a transit domain, respectively).

An important assumption is that domains can be label producers, label consumers or both - these represent inherent properties of a domain. Thus negotiating a stitching operation for a slice requires deciding on which domains will produce and which will consume labels, a process that is constrained by the topology of the slice and the capabilities of the domains (i.e. domains that can only consume labels cannot be expected to produce them). This problem resembles the 'dining philosophers' problem, as one of the aspects of the negotiation necessitates freedom from deadlocks between e.g. two label producing domains. In ORCA operations on domains during slice creation are done as part of the 'redeem' operation, thus, in effect, what is required is determining the order in which multiple tickets from the user for components of the slice are redeemed from different domains.

There are multiple approaches to this negotiation process. If we simplify our view of a slice to a single path connecting resources across multiple transit domains and make an assumption that this inter-domain path has been computed, then our options for negotiating labels across domain boundaries can be enumerated as follows:

- *Hop-by-hop stitching* In this approach domains local setup and produce or consume labels from their neighbors. This process is usually assisted by signaling which is done in two passes, as e.g. taken by GMPLS family of standards, with the stitching done on the reverse pass using the RESV message. The main disadvantage of the approach is difficulty of dealing with deadlocks in a generalized environment such as the one we describe. It also incurs the longest provisioning delay as the signaling message needs to go over each domain sequentially, waiting for the completion of intra-domain provisioning before proceeding to the next domain.
- *Centralized stitching* In this approach, a centralized entity computes the labels for every domain on the path and sends a request to each domain simultaneously to complete the configuration. The primary disadvantage is the lack of control by the domains and the single point

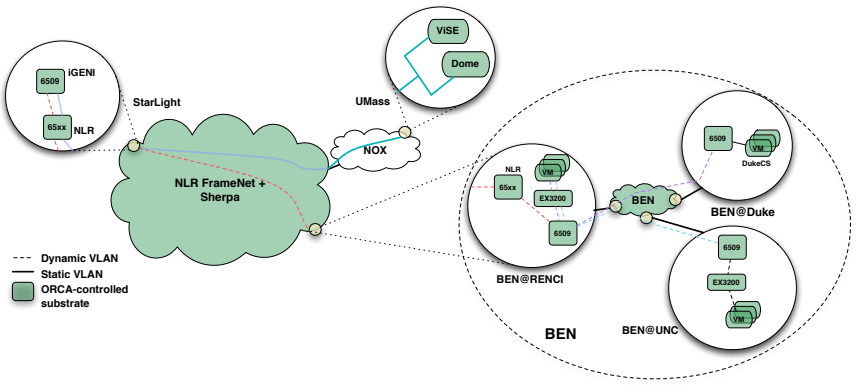

Fig. 2. ORCA demo topology

of failure presented by this function.

- *Coordinated stitching* In this approach, each domain reports limited capability information to a coordinating entity (a number of which can exist). The coordinating entity makes a decision about the order of redeem operations and informs the domains. The domains perform the resource allocation (in this case VLAN provisioning) in specified order and inform their neighbors (if they are label producers) or wait for their neighbors to complete (if they are label consumers).

We have adopted the third option, as allowing the most flexibility for future expansion. The coordinating entity decides the order in which the user tickets are presented/'redeemed' with the different sites, the sites either report produced labels or consume labels reported by other sites. Note that the process of ordering these redeem operations translates into creating multiple partial ordered sequences of domains in which label producing and consuming operations alternate between adjacent domains. In our implementation, the coordinating entity computes a domain dependency graph (which must be a DAG to avoid deadlocks) of domain producer/consumer relationships. For a simple path, this DAG looks like a 'forest' with the root of each tree representing a domain that is not dependent on any other domains (a label producer). The domains at the roots of the trees are the first to honor the redeems (i.e. perform their site resource allocations). The coordinator then follows the DAG to arrange the redeems in other domains, passing the label information down, until the path is complete and a slice is stitched together.

Our current heuristic for computing dependency graphs for a single path takes into account a number of domain attributes, listed here in the order of decreasing priority:

1) Label type: Lower layers in the network are dealt with first, due to the aforementioned client-server relationship between the network layers.
2) Producer vs. consumer: Producers take precedence
3) Label translation capability: Domains not capable of label translation are given precedence
4) Fixed peering relationship: Domains may have existing fixed labels connecting them to neighbor domains, which removes the need to negotiate labels
5) Nodal degree: Higher nodal degree domains are given precedence

Computing these dependency graphs for generalized topologies is an area of active investigation for this project.

V. Project Status and Future

We have deployed the ORCA software prototype in the BEN network testbed and at other sites (see Figure 2). BEN is a metro-scale dark fiber facility with several PoPs (Points of Presence) in North Carolina's Research Triangle. Each PoP in a BEN network is equipped with a fiber switch, a DWDM gear from Infinera supporting OTN and a L2/L3 switch/router from Cisco. When creating slices, ORCA uses native TL1 and CLI interfaces of all of these network elements to create appropriate crossconnects or circuits to support the multi-layered topologies needed to create L2 VLAN connections across BEN. At the edge of BEN we have deployed a number of Eucalyptus [19] clouds exposing Amazon EC2 interfaces for creating VMs on specific VLANs that ORCA can use to instantiate VMs. ORCA also has an interface to NLR Sherpa [20] tool - a CGI-based dynamic VLAN provisioning service offered by NLR. Each of these domains (Eucalyptus domains, BEN and NLR domains) have an ORCA actor responsible for their configuration. Through the collaboration within GENI we also have ORCA actors deployed at the StarLight facility in Chicago, where ORCA controls a L2 switch, as well as in UMass Amherst, where ORCA controls a weather sensor testbed called ViSE [21].

The BEN and NLR sites expose a range of available VLAN tags to an ORCA broker. NLR site is a 'label producer' (it generates a VLAN tag for the requested circuit), while BEN site serves as both producer and consumer (it can accept and translate incoming tags). Eucalyptus and testbed sites expose their resources to the broker using NDL-OWL descriptions. The user requests a specific slice configuration through a GUI that generates NDL-OWL (in RDF-XML format) of the request that is then processed by an ORCA agent to compute the inter-domain path and the dependency graph. The agent then requests tickets for necessary resources from the broker and redeems them in the proper order to ensure slice stitching. The sequence of redeems is shown in the timing diagram in Figure 3. Instantiated VMs boot with two interfaces - one

for experimenter access over commodity Internet, the other connected to the slice VLAN to facilitate intra-slice traffic. Network elements are configured by the site actors responsible for them. A private-address IP network is configured inside this Layer 2 slice.

The particular experiment demonstrated at the GEC7 Conference in March of 2010 creates an end-to-end slice by instantiating a VM instance at the Eucalyptus cloud site on Duke University campus and a Xen VM from the ViSE testbed; it links them through the dynamically stitched VLANs from several providers. It then launches an Apache Web server on its Eucalyptus node, which provides an interface to process and visualize radar data fed through the circuit from the Xen node at UMass. The total slice instantiation time is approximately 4 minutes.

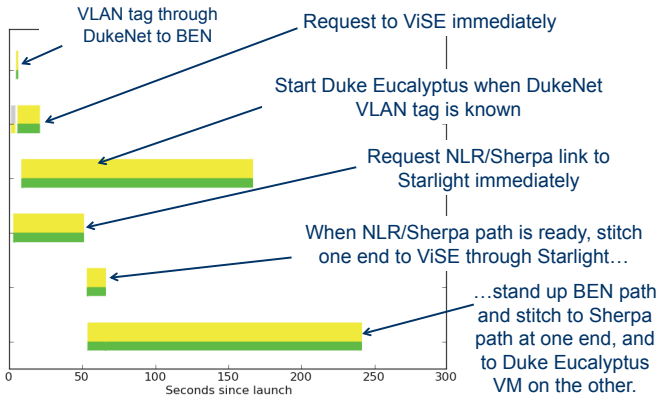

Fig. 3. Timing diagram showing redeem sequence

Our initial experience with the ontology-based approach has been promising: the prototype demonstrates policies for reliable dynamic provisioning of multiple, concurrent, isolated slices of the BEN network, in tandem with Eucalyptus virtual machines provisioned from the edge sites, all utilizing semantic resource descriptions. We are extending our work to to enrich the abstract view of the BEN network to include more layers and edge resources. At the same time we are working on developing new resource allocation policies with the focus on embedding random topologies into the available substrate.

VI. Conclusion

This paper summarizes our experience in designing and implementing a *meta*-control framework for managing distributed resources, capable of creating orchestrated connected configurations of these resources on demand. The control framework is free of any assumptions of the type of underlying substrates and our goal is to build a general operational system that will support many different types of resources and serve many application domains. Our approach is enabled by powerful architectural abstractions, a distributed implementation and a declarative ontology-based resource description mechanism. Our demo serves as a proof of concept to show how applications can interconnect and link resources through dynamic circuits provisioned along with the VMs by a cloud orchestration framework.

Acknowledgement. This work is supported by the National Science Foundation GENI Initiative, NSF awards CNS-0720829 and CNS-0910653, and an IBM Faculty Award. The authors wish to thank Joe Mambretti for assistance with deploying and maintaining experimental equipment in the StarLight facility at Northwestern University, Chicago IL.

References

[1] M. A. Vouk, "Cloud computing - issues, research and implementations," *Journal of Computing and Information Technology*, vol. 16, no. 4, Dec. 2008.

[2] Y. Wu, M. C. Tugurlan, and G. Allen, "Advance reservations: a theoretical and practical comparison of gur & harc," in *MG '08: Proceedings of the 15th ACM Mardi Gras conference*. New York, NY, USA: ACM, 2008, pp. 1–1.

[3] G. Zervas, "Phosphorus Grid-Enabled GMPLS Control Plane (G2MPLS): Architectures, Services, and Interfaces," in *IEEE Communications Magazine*, Aug. 2008.

[4] S. Thorpe., L. Battestilli, G. Karmous-Edwards, A. Hutanu, J. MacLaren, J. Mambretti, J. Moore, S. Sundar, Y. Xin, A. Takefusa, M. Hayashi, A. Hirano, S. Okamoto, T. Kudoh, T. Miyamoto, Y. Tsukishima, T. Otani, H. Nakada, H. Tanaka, A. Taniguchi, Y. Sameshima, and M. Jinno, "G-lambda and EnLIGHTened: wrapped in middleware co-allocating compute and network resources across Japan and the US," in *GridNets '07: Proceedings of the first international conference on Networks for grid applications*, 2007.

[5] I. Baldine, J. Chase, G. Rouskas, and R. Dutta, "At-scale experimentation with resource virtualization in a metro optical testbed," in *Proceedings of ICVCI, 2008*, May 2008.

[6] D. Irwin, J. S. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum, "Sharing Networked Resources with Brokered Leases," in *Proceedings of the USENIX Technical Conference*, June 2006.

[7] J. Chase, L. Grit, D. Irwin, V. Marupadi, P. Shivam, and A. Yumerefendi, "Beyond Virtual Data Centers: Toward an Open Resource Control Architecture," in *Selected Papers from the International Conference on the Virtual Computing Initiative (ACM Digital Library)*, May 2007.

[8] A. Yumerefendi, P. Shivam, D. Irwin, P. Gunda, L. Grit, A. Demberel, J. Chase, and S. Babu, "Towards an Autonomic Computing Testbed," in *Workshop on Hot Topics in Autonomic Computing (HotAC)*, June 2007.

[9] J. Chase, I. Constandache, A. Demberel, L. Grit, V. Marupadi, M. Sayler, and A. Yumerefendi, "Controlling Dynamic Guests in a Virtual Computing Utility," in *International Conference on the Virtual Computing Initiative (an IBM-sponsored workshop)*, May 2008.

[10] I. Constandache, A. Yumerefendi, and J. Chase, "Secure Control of Portable Images in a Virtual Computing Utility," in *First Workshop on Virtual Machine Security (VMSec)*, October 2008.

[11] H. Lim, S. Babu, J. Chase, and S. Parekh, "Automated Control in Cloud Computing: Challenges and Opportunities," in *Proc. of the First Workshop on Automated Control for Datacenters and Clouds (ACDC)*, Jun. 2009.

[12] T. Berners-Lee, "Design issues: Architectural and philosophical points," in *W3C*, 2008.

[13] W3, "Resource description framework (RDF)," http://www.w3.org/RDF/.

[14] J. Ham, F. Dijkstra, P. Grosso, R. Pol, A. Toonk, and C. Laat, "A distributed topology information system for optical networks based on the semantic web," *Journal of Optical Switching and Networking*, vol. 5, no. 2-3, June 2008.

[15] ITU-T. G.805 : Generic functional architecture of transport networks. [Online]. Available: http://www.itu.int/rec/T-REC-G.805

[16] F. Kuipers and F. Dijkstra, "A path finding implementation for multi-layer networks," *Future Generation of Computing*, vol. 25, no. 2, Feb. 2009.

[17] "Global lambda integrated facility (glif)," http://www.glif.is/.

[18] F. Dijkstra, "Framework for path finding in multi-layer transport networks," 2009.

[19] "Eucalyptus," `http://eucalyptus.cs.ucsb.edu/`.

[20] "NLRFrameNet Dynamic VLAN Services/Sherpa," http://globalnoc.iu.edu/nlr/maps_documentation/nlr-framenet-documentation.html.

[21] D. Irwin, N. Sharma, P. Shenoy, and M. Zink, "Towards a virtualized sensing environment," in *Proceedings of the 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, May 2010.