Eucalyptus: A Web Service-Enabled E-Infrastructure

Liu, Sandy; Liang, Y.; Brooks, Martin

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Eucalyptus: Provisioning Resources on e-Infrastructure **

Liu, S., Liang, Y., Brooks, M.
October 2007

* published at The 17th Annual International Conference on Computer Science and Software Engineering (CASCON 2007). October 22-25, 2007. Toronto Ontario, Canada. NRC 49847.

Canada

Eucalyptus: A Web Service-Enabled E-Infrastructure

Sandy Liu, Yong Liang, Martin Brooks

email: {Firstname.Lastname}@nrc.gc.ca

Institute for Information Technology, National Research Council Canada

Abstract

With the support of user configurable high speed networks, the emerging e-Infrastructure allows seamless sharing of expensive scientific resources. These resources are often running on a variety of platforms, have different bandwidth and QoS requirements, require specific configuration by technical experts, and in most cases cannot be accessed through a single point of entry. To address these issues, we propose an extensible, reliable, and simple software architecture to share the applications and resources over hybrid networks, and hide the tools' logistical and provisioning complexities.

This paper explores the design and implementation of Eucalyptus, and describes how it leverages the benefits of a Service-oriented Architecture (SoA) to provide a highly adaptable, modular, and loosely coupled solution to configure and manage resources needed by users collaborating over the net. We present our methodology to wrap functions of resources into Web services, and integrate the new Web services into the Eucalyptus platform in a generic way. The streams of the events from these resources are captured. This information is used for monitoring resources' activities and diagnosing any error that may arise. We provide a workflow management service allowing users to orchestrate services based on the description of the resources, their dependencies and the captured streams to perform certain tasks. We also propose a combination of Web services and peer to peer technologies to support users in different communities and different network layers, and to decentralize resource management. Eucalyptus was demonstrated to be effective in assisting architects across multiple sites to effectively participate in a shared design session.

1 Introduction

E-Infrastructure is emerging with advances in sharing expensive and data intensive resources over high speed networks. These resources are often running on a variety of platforms, have different bandwidth and QoS requirements, require specific configuration by technical experts, and in most cases cannot be accessed through a single point of entry. To address these issues, we propose an extensible and simple Web Service based middleware to allow on-demand provisioning of software applications, device, tools, and its underlying networks (collectively called *resources*), thus hiding the tools' logistical and provisioning complexities.

Eucalyptus takes a simple but general approach to computer-supported collaboration. Some collaboration systems expect users to learn new tools and adapt themselves to a specific regimen that induces sound collaborative practices. Eucalyptus, on the other hand, provides an infrastructure that allows the user's to share with each other the resources that they are already familiar with. These resources include applications, data and services.

A Eucalyptus session is a set of resources and people making use of those resources at the same time. For instance, one session might include a videoconference system that joins two rooms with

one person in each room, and a third person in a different location without video equipment; meanwhile all are contributing to a shared digital whiteboard. Another example session includes users in different locations accessing a common chat room, while one of the users is sharing his desktop with the other users via a desktop sharing application that allows the other users to view his desktop from their computers. For a third example, a pair of users are sharing a desktop and collaborating on a digital description of a three-dimensional scene; a rendering computer is available in the background that converts the scene to a true-to-life images and sends the files to all of the participants. During the course of the collaboration, the scene is repeatedly rendered, viewed and adjusted until the participants are satisfied. Thus we can see that sessions can make use of a variety of resources, including various applications, devices, and files.

By simultaneously delivering any combination of these resources, Eucalyptus sessions add value to many of the groupware applications that are found on the internet: online chat, videoconferences, data sharing, and application sharing. Eucalyptus adds more value by providing two other features. First it provides users with the ability to build workflows to script the basic steps in provisioning these resources. Second it allows users to schedule the provisioning of these resources so that a meeting can be set up ahead of time and delivered with a single click.

Eucalyptus is integrated with optical network provisioning through user controlled lightpaths [7]. Once a user selects the resource and its corresponding participants, Eucalyptus can compose a workflow that includes the adaptation of the logical network topology. This combination means that an entirely new type of videoconference is available to the end user: on-demand uncompressed high-definition two-way videoconferencing. Uncompressed high definition videoconferencing gives an unsurpassed experience. Because the data is transmitted over high-capacity lightpaths, it does not need to be compressed, so there is almost no latency in sound or picture. Furthermore because the signal is transmitted over a dedicated, direct connection, the bandwidth is constant. It can support multiple gigabits per second of transmission, which means that the end user is offered a high definition, jitter-free, delay-free experience. Moreover the high bandwidth network does not need to be dedicated to the resources outside of the time that the resources are using them. The network is in fact one of the resources that can be made available for just the time it is needed, using the UCLP software interface.

In the subsequent section, we briefly introduce the concepts of hybrid networks and the notion of user controlled lightpaths. We then provide an overview of the system design, our current implementation, demonstration experiments and follow this by the related work and conclusion.

2 Background

2.1 Web Services-based Service-oriented Architecture (SoA)

Service-oriented Architecture (SoA) is the latest software architecture style to build flexible and extensible applications. OASIS describes SoA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [18]. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. Web Services represents a set of *de facto* SoA implementation technologies, involving the usage of W3C standards such as WSDL [5] and SOAP [4] for service description and messaging transport respectively. Web Service's component-based, web-oriented, standard-based, language, platform, and domain independent nature makes it an appropriate solution for many system and data integration projects. We adopt this approach for provisioning resources spanning from networks to devices.

2.2 Hybrid Network and Articulated Private Network

With the exponential growth of the Internet and the increased cost of routing, the layer 3 (IP network) sometimes cannot provide the required bandwidth and stability required by certain applications. Many e-science projects involve the usage of remote sensors and instruments generating very large volumes of data that need to be delivered and processed in far away facilities. Similar situation for architects and industrial designers, who need to share high quality multimedia files in

real-time. This calls for high transport capacity networks.

A hybrid network provides a practical solution. It consists of both the traditional routed IP access (layer 3) to the Internet and circuit switched point-to-point connections (layer 2). These connections are often referred to as *lightpaths*. More specifically, a *lightpath* is an abstraction of a connection between two or more switches in an optical network, and typically connects two points on the network at speeds up to 10 gigabits per second. These lightpath connections offer a guaranteed Quality of Service (QoS) regarding bandwidth and latency.

To fulfill the demand for network bandwidth and QoS, advanced network organizations such as CANARIE[1] have been investigating ways to provide application-oriented and user-controlled networks services. A resulting product is called the User-Controlled Lightpath Provisioning (UCLP) [7] tool. UCLP is a Web Services based solution for provisioning lightpaths. UCLP can be thought of as a configuration and partition manager that exposes each lightpath in a physical network and each network element associated with a lightpath as an "object" or "service" that can be put under the control of different network users to create their own logical IP network topologies [14]. The network users can then reconfigure and partition the lightpaths. This privately articulated end-to-end network is therefore called Articulated Private Network (APN). Within each APN, a number of network scenarios (i.e. logical topologies) can be specified to support different applications and usage scenarios. The APN Web Service can then be generated as a BPEL (Business Process Execution Language) workflow linking together various network elements across multi-domain networks. When an APN BPEL workflow is deployed and published, the applications can set up the suitable network topology by invoking the APN Web Service.

However, these high-speed connections are not pervasive, often only a limited end-points are connected through lightpaths. To leverage the benefit of a hybrid network, we configure gateway computers that have access to both routed IP networks and the switched lightpath networks. We then deploy the set of management Web Services on these gateway computers. As all available resources are published and maintained through these management services, consequently users of Eucalyptus can conveniently look up resources via a layer 2 or layer 3 connection, and provision the resources (including the underlying networks) effectively through corresponding Web Services.

[1] A non-profit organization who provides a national optical Internet research and education network in Canada. h*ttp://www.canarie.ca/canet4/index.html*.

3 Overall System Design

Most functions in Eucalyptus are provided by Web Services, either as a single service or a combination of services. We divide the services into three groups: primitive resource-oriented services, management services, and utility services as shown in Figure 1. Eucalyptus can be seem as a control panel for a set of network accessible resources, being locally or remotely available. This control panel is represented as a light-weight dashboard application sitting on the desktop of a user's computer. Through this control panel, one can define what resource is needed, when, where, and by whom, therefore providing a description of a user-defined session. The system can then generate a session described as a Web Service workflow involving services to reconfigure the network if necessary and to launch the corresponding resources with the proper set of parameter values at the time specified.

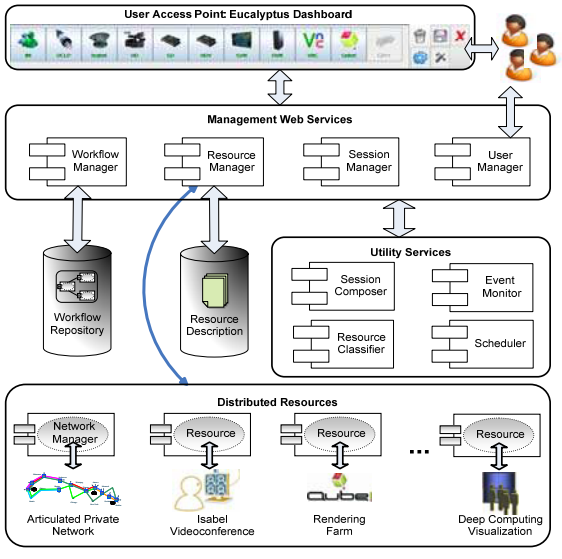

Figure 1: Eucalyptus System Overview

Eucalyptus has a large community of users and needs to provision different resources hosted by Web services running on different platforms. Management of Web services and resources is a crucial part of Eucalyptus. To monitor and manage the resources, the management service should know the following information [10]:

1. The identity of all resources, which includes the resources properties, such as name, category, permission, etc.
2. Resources host, which includes the Web service platform, version number, etc.
3. Whether the Web service is working well and the resource is functioning, and the status of the resources
4. The average request and response time of invoking a given resource
5. The average message size when invoking a given resource
6. The workload of a given resource in terms of calls per hour

Thus we created an Event Monitor aiming to capture some of this information and broadcast these events to the related management services. In Eucalyptus, we use a set of central Web services to conduct the management tasks as shown in Figure 1. One important factor for a central management service is that Eucalyptus sits on different network layers. A Web service from a layer 2 network typically can not communicate directly with a layer 3 Web Service. We use a central management service as a mediator to direct the user's requests to the destination resource-oriented Web service regardless of which network the user is signing in from, being layer 2 or layer 3. The central management services make it easy to capture all information for measuring the performance of Eucalyptus, and reduce the cost of bringing all events together. Beside the dashboard GUI, we plan to provide a management console to help the admin users to manage Eucalyptus.

The rest of this section outlines how each individual component works and the interactions among these components.

3.1 Resource Management

Every resource is governed by a Resource Manager in Eucalyptus and each resource should provide a Web Service interface for the resource manager to provision it. The basic provisioning tasks for an individual resource include launching, shutting down, or checking the status of resources. Note that we are only concerned about provisioning of the resource, not the actual data communication among resources. For example, to launch a multi-point video-conference application, Eucalyptus starts the conference application with the proper parameters and configures the underlying network to make sure it can support the bandwidth requirement. The actual communication among different conference machines is handled by the native application.

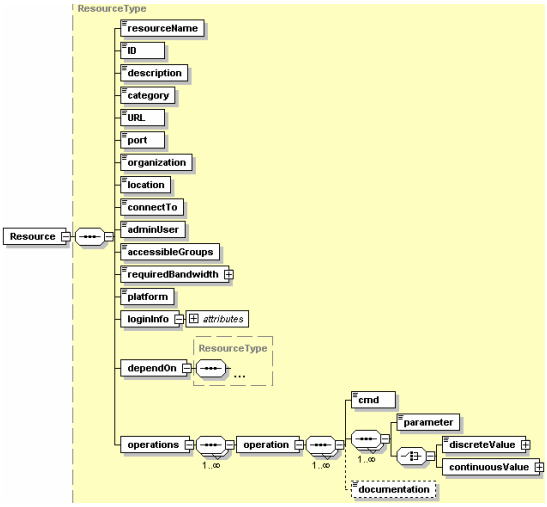

Figure 2: Resource Schema

To allow new resources to be easily integrated into Eucalyptus, we provide a generic approach for wrapping up a resource and making it accessible through Web Service. We define each non-network resource with an XML description file. Each resource is assigned a Resource ID and is described by a set of non-functional descriptions, including name, category (e.g. communication tools, visualization tools, etc.), physical location, URL, port, the admin user, the access restrictions (which user group has access to this resource), what platform is this resource running on (i.e. Windows, Linux), the login information for accessing the machine, which router or switch it is connected to, the bandwidth requirement, and any resources it depends upon. In addition, the XML file also specifies the operations supported by this resource; each operation is described by a command and the corresponding parameters. Figure 2 shows the schema for defining resources.

```
Class(pds:Resource partial
   unionOf(
     pds:CommunicationTool
pds:VisualizationTools
     pds:ManagementTools
pds:OtherResources
   )
   restriction(
     pds:requires allValues-
From(pds:Resource)
   )
 )

ObjectProperty(pds:requires)

Class(pds:VideoConference complete
   pds:Isabel pds:UCLPVideoConference)
 )

Class(pds:HighSpeedNetwork partial
   unionOf(pds:UCLPNetwork)
 )

Class(pds:InstantMessenger complete
   pds:EucalyptusIM
 )

Class(pds:UCLPVideoConference partial
   restriction(pds:requires
     someValuesFrom(pds:UCLPNetwork)
   )
 )
```

Figure 3: Semantic Description of Resource in OWL

All the descriptions are stored in the Resource Description Database. We also provide a Resource Classifier, With the increased number of resources being added to Eucalyptus, users may want the system to choose a resource that is most appropriate at a given time for a given task; or when a resource breaks down, the system can provide a list of alternative resources with similar capabilities. To support this, we group resources into different categories, represented as a class hierarchy. Each class is distinguished by a set of properties that differentiate its capabilities. The knowledge about classes and properties can be represented conveniently by description logic. Figure 3 shows a snippet of the description in OWL [3] (using concrete abstract syntax [11]), a Web Ontology Language based on description logic. The first axiom states that the set of resources includes tools for communication, visualization, management and other resources. It also states that a resource may require another resource (the dependency relationship). The final axiom states that all the UCLP video-conference tools require an underlying UCLP network. This semantic description is stored in the Resource Classifier as a knowledge base. The resource classifier runs a rule engine, where one can query about the properties of a resource, and the type of resources that falls under the same class (i.e. same category). This is particularly helpful when a requested resource becomes unavailable, the system can make a query to the resource classifier to identify if an alternative resource is available.

Equipped with the resource description and the semantic information for categorization, Eucalyptus can use the resource wrapping utility to quickly generate the corresponding Web Service. For adding a new instance of an existing type of resource, we can simplify deploy the Web Service of the same type to the machine that hosts that resource-oriented Web Service. For instance, if a system already have a DCV (Deep Computing Visulization) Web service for visualization in Eucalyptus, and a user wants to share his/her new DCV resource, Eucalyptus can generate a new DCV Web service from the existing template and deploy it to the Web service platform provided by the user. Eucalyptus can then take in the WSDL description of the new resources and automatically generates the client codes for invoking the Web service. The Resource Management WS will call the generated codes dynamically at runtime to invoke the newly integrated resource through its own Web Service. The approach is also applicable when a user wants to add a resource that has already been wrapped as a Web Service.

The resources in each category have similar functions, which means every resource in the same category has similar operations to be exposed to the user. For example, in communication tools, all resources should have the following functions: **startResource(), stopResource(), getStatus()**. Different resources have different input and output parameters in their functions. To make the generic Web Service interface extensible to all of the more specific resource types, we declare all the input parameters and the return type as **String**. We will use parameter information described in the XML resource description file to parse the input and output strings properly.

3.2 User Management

All users of Eucalyptus are managed by the User Manager, who keeps the detailed users' information, and assigns the users into different groups.

It is also responsible for checking the users' credential for authentication and authorization. Each Eucalyptus user will need to be identified upon login to the system. The user manager keeps track of a user's profile including user name, login ID, password, contact information, associated organization, preference, current login location, usage history, and the user group s/he belongs to, which ultimately determines his/her access restriction for the resources. The User Manager works with the Session Manager to determine if the user has the access rights to a certain resource.

3.3 Session Management

We try to hide the dependencies among the resources and combine the session semi-automatically. A Eucalyptus session, defines a task involving certain resources and the sequence of activities related to the resources. Although most of the resources are wrapped as Web services, a session is more than a Web service composition. For example, a user could define a videoconference session for a large group of people that 1) involves four Web services; 2) lasts for 2 hours; 3) involves different resources in the middle of the session for sub-groups discussions; 4) checks the status of the resources, records the status, and reflects this information back to the user. Such sessions are described in XML. The XML input can be translated automatically into two BPEL processes as shown in Figure 5. Figure 6 shows the basic process for every session, the Session Management Web Service (SMWS, also known as Session Manager) calls the User Management Web Service(UMWS, also known as User Manager), and Resource Management Web Service (RMWS, also known as Resource Manager) to check the authority of the users and the availability of the resources. In Figure 6, a Web service is started for monitoring. It checks the status of the running resources, and times the process. A pick activity blocks the thread of execution until a message event or alarm event occurs. Some basic events are listed as follows:

```
<session>
 <description>
   A broadband videoconference session
 </description>
 <sequnce>
   <session>
      <description>
         start an Isabel connection and a Pleora connection
      </description>
      <and>
      <resource>
         <id>Isabel003</id>
         <category>communication tool</category>
         <name>Isabel</name>
      <location>nrc-ottawa</location>
         <connectto>Isabel003</connectto>
         <starttime>1100</starttime>
         <endtime>1115</endtime>
      </resource>
      <resource>
         <id>Isabel0034</id>
         <category>communication tool</category>
         <name>Isabel</name>
      <location>nrc-nb</location>
         <connectto>Isabel004</connectto>
         <starttime>1100</starttime>
         <endtime>1115</endtime>
      </resource>
      </and>
      <and>
      <resource>
         <id>Pleora003</id>
         <category>communication tool</category>
         <name>Pleor</name>
      <location>nrc-ottawa</location>
         <sendto>Pleora004</sendto>
         <receivefrom>Pleora004</receivefrom>
         <starttime>1100</starttime>
         <endtime>1115</endtime>
      </resource>
      <resource>
         <id>Pleora004</id>
         <category>communication tool</category>
         <name>Pleor</name>
      <location>nrc-nb</location>
         <sendto>Pleora003</sendto>
         <receivefrom>Pleora003</receivefrom>
         <starttime>1100</starttime>
         <endtime>1115</endtime>
      </resource>
      </and>
   </session>
   <wait>PT300S</wait>

   <session>
     <description>
       start a Pleora session
     </description>
      <and>
      <resource>
         <id>Pleora003</id>
         <category>communication tool</category>
         <name>Pleor</name>
      <location>nrc-ottawa</location>
         <sendto>Pleora004</sendto>
         <receivefrom>Pleora004</receivefrom>
         <starttime>1120</starttime>
         <endtime>1145</endtime>
      </resource>
      <resource>
         <id>Pleora004</id>
         <category>communication tool</category>
         <name>Pleor</name>
      <location>nrc-nb</location>
         <sendto>Pleora003</sendto>
         <receivefrom>Pleora003</receivefrom>
         <starttime>1120</starttime>
         <endtime>1145</endtime>
      </resource>
      </and>
   </session>
 </sequnce>
</session>
```

Figure 4: A Session Description

A time out alarm An activity will be executed to call a daemon program. The daemon callbacks the client to ask whether the conference is finished or not and waits for the user's response, which causes a stop message event to occur.

A stop message An activity will be executed to invoke the corresponding Web service to stop the resources.

A resource fails message An activity will be executed to check whether there are other resources available. If not, the process terminates with a fault; otherwise, we consider the process completed.

3.4 Workflow Management

As illustrated above, a session definition defines the workflow of how different resources and users interact and the sequence of execution. This information can be saved as a workflow in the workflow repository for future retrieval. These workflows are managed by the workflow manager, who is responsible for the insertion, deletion, and modification, and execution of workflows. Whenever the session manager receives any interruption events from the Event Monitor, it passes through to the Workflow Manager to withdraw the execution of the workflow and execute some compensation activities. The session manager will communicate with the user for future actions. In some cases, it may re-compose a new workflow as the alternative. For example, when a high speed broadband connection becomes unavailable, the high-definition videoconference session will be interrupted, however, the session manager may suggest the session host to resume the session with a standard definition videoconference resource such as a Pleora SD session or an Isabel session, both can run without a broadband connection.

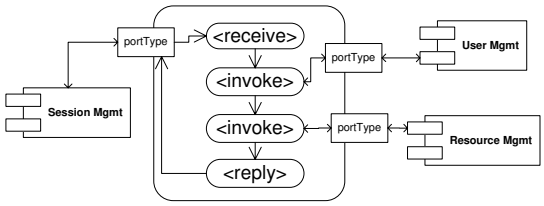

Figure 5: Session Management Web Service

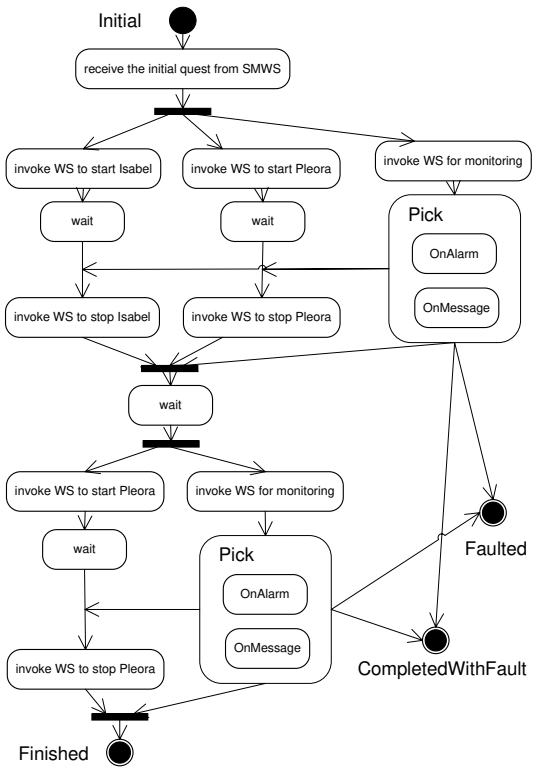

Figure 6: Basic Process of A Session

3.5 Dashboard: The Control Panel

In order to provide a convenient single entry point for users to access all the available resources in Eucalyptus, we provide a simple dashboard as a control panel for users to define and schedule sessions with the desired properties. This graphical user interface acts as an integrated Web Services client that hides the complexities of configuring the resources required, providing access to those resources through a few clicks of buttons. The dashboard is developed as a desktop application as opposed to a web application for several reasons: 1) Web applications have limited functionality on the client computer. There is no easy way to access the local file systems. 2) The implementations of the HTML, CSS, DOM and some other tools are browser specific and they often act inconsistently in different browsers. 3) It is less convenient to maintain an accurate reflection of status of all the resources with a web application since it does not have its own thread.

To maintain a desktop application over many computers is normally not an easy task. However,

with the help of Java Web Start [16], the deployment and maintenance of Java desktop applications becomes easier. The advantages of Java Web Start include automatic application update, desktop integration, platform independence, Java runtime environment management, and security.

The dashboard interface is carefully designed to be unobtrusive and user-friendly. Inspired by DragThing [21], it is implemented as a floating dock, similar to the Mac OS X system dock. The dashboard (sometimes also referred to as the FloatingDock) only appears at the bottom of the desktop, and it can be anchored to any other edge of the desktop.

Figure 7: The Eucalyptus Dashboard

Each resource has its own button on the dashboard as shown in Figure 7. The user can also define his/her own often-used resources on the dashboard and can add them as new buttons. In addition, an existing executable application can be dragged and snapped into the dashboard, as a shortcut to launch that application.

3.6 Peer-to-Peer (P2P) Eucalyptus

Figure 1 shows how Eucalyptus works in one management domain, i.e. all the resources are being managed by one set of management services. This can post two main corners. First, the reliability and availability of the management service. Since most of the management-related Web Services can be essentially hosted by one physical node on a network. If this node breaks down, the user will not be able to get any access to the resources registered with the Resource Manager. Second, there are situations where an organization may not want to expose all the resources and users to another organization, who runs the management services. To resolve the above mentioned problems, we propose a hybrid Peer-to-Peer(P2P) Eucalyptus management architecture where each *Eucalyptus Peer* has its own set of management services, user groups, and resources. One of these peers is designated as the default host that keeps track of a list of registered peers. This list is shared by the peers as well as the dashboard. Each peer maintains a cache of the list and it connects to the default host for updates at a set frequency (e.g. every hour). Each peer maintains a list of available peer by sending a ping to each listed peer. In case the default host becomes unavailable, the next peer appearing on the list will become the default host. The new list will be propagated to other peers. Figure 8 illustrates a system with 5 active peers.

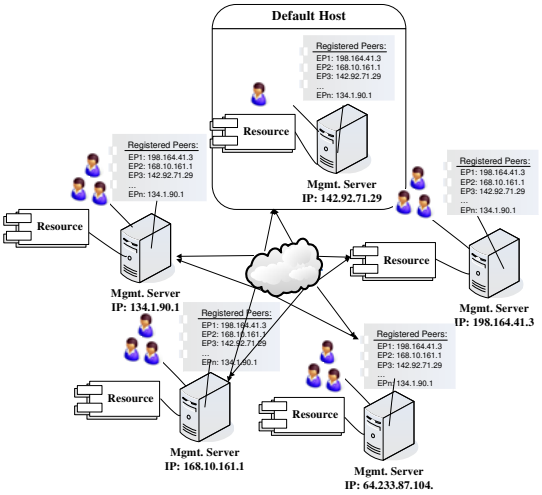

Figure 8: P2P Eucalyptus Architecture

4 Current Implementation and Experiments

The current Implementation of Eucalyptus is being used for geographically distributed architects and industrial designers to use in their participatory design sessions. Thus most resources are selected to support architectural design activities. This includes some high-end devices that are not easily available in most labs, for example, the IBM Deep Computing Visualization cluster, ren-

dering farms, high-definition videoconferencing devices, etc.

We have conducted serveral live demos to showcase the concept of Eucalyptus and to test the capability of the system. One of the demo shows Eucalyptus running on the mixed layer 3 and 4Gb lightpath environment connecting Carleton Immersive Media Studio (CIMS), the Society for Arts and Technology (SAT) in Montreal, and the Communication Research Centre Canada (CRC) in Ottawa. We deployed the management Web Services on a server (called Service Manager) at CRC, which is configured to be on both layer 2 and layer 3 networks. The first demo scenario shows the architects located at CIMS and SAT use the Eucalyptus interface to control visual communication and 3D modeling tools in their participatory design session. They use the IBM's Deep Visualization Computing (DCV) [1] cluster technology to provide hands-on direct collaboration using Maya. With DCV, high-end graphical images can be created in a visualization mode that distributes graphical images to remote (collaborative) clients. With Eucalyptus, the architects do not need to know each single configuration parameters and deal with the Linux server (that DCV is running on) or the system administrator before using this kind of tools.

While working on the design by sharing the desktop, the designers may want to see and hear each other for more discussions. Eucalyptus provides the access to some videoconference tools and they can be configured and launched automatically by Eucalyptus according to user's preference. For instance, one can choose the uncompressed Standard Definition (SD) video (480i) - DVD-quality streaming at 300Mb using Pleora technology. When graphically precise work details at a designer's desk need to be shared, Pleora HDV (720p) close-up view gives startling clarity. When larger-scale views must be shared at full resolution, for example when viewing street-level activity, UltraGrid [20] uncompressed HD (1080i) provides the highest resolution video. Eucalyptus not only do provide easy access to these conference tools, but also provide the corresponding network setup as a workflow for the underlying network connections.

Now the school of architecture in Pennsylvania State University, and the Carleton University (Canada) are using Eucalyptus in their collaborative project. This involves about 30 students from both schools forming teams of four in co-designing an aviation museum.

A highly interactive and novel break-dance session was held between Carleton and Society for Arts and Technology (SAT) in Montreal on 8 March 2007. Using uncompressed high-def and standard-def two-way video on a 2 Gb/s lightpath, break dancers at each site challenged each other to successively more difficult moves and routines. The dancers from separate locations were able to interact and stay synchronized. We have also experienced a real-time dance performance of the Korean Nulhui Dance Company, simultaneously transported from Seoul, Korea and Ottawa, Canada and to Barcelona, Spain [17]. For our applications, the "uncontested bandwidth" feature is the biggest advantage of lightpaths. The ability to bond multiple one-gigabit lightpaths into a single multiple-gigabit lightpath is also an advantage.

5 Related Work

Contract-first development is a new approach for developing Web services, the idea of which is to define the contract, such as data, operations and messages first, and then generate the appropriate code. In Eucalyptus, we apply this method in many places. For example, when a new Web service comes in, we generate the WSDL from the XML schema [2], and then generate the client code for invoking this Web service; based on the user's requirement and contract WSDL, we generate the workflow represented as a BPEL file. However, most parts in Eucalyptus are developed using a traditional code first method. We found that it is difficult to define the formal contract at the beginning. Initially Eucalyptus is designed for the architects, and the user's requirements are not well defined, so we use a spiral approach: develop, testing, deploy for authentic use, and revise. In addition, at this point there is no single, unified development environment for contract-first development, and most of the available tools are developed separately [12]. We will continue to develop Eucalyptus in this hybrid approach. For the resource-oriented Web Services where many developers need to implement and deploy their new Web services, we use the contract-first development to improve the interoperability, while for the management and utility Web Services, we use the code-first technique.

There are some other message-based Web Services integration, selection, and management

techniques that adopt the concept of Enterprise Service Bus for developing, deploying and monitoring integration projects across the enterprise [19]. The Web Service Management Layer [13] also provides a central management layer for dynamically integration, selection and composition by putting all Web services related client code into the management layer. WSIF [6] is a WSDL based API for invoking WSDL based Web services no matter how the Web services are implemented. It provides a generic client framework which unaware of the service update, migration and the change of protocols. Although current implementation of Eucalyptus wraps all resources in Web services based on SOAP API under Apache AXIS, we consider employing the WSIF's API in future implementations to better incorporate Eucalyptus with other legacy applications such as EJB, JMS applications.

6 Conclusion and Future Work

In conclusion, Web Service's component-based, web-oriented, standard-based, language, platform, and domain independent nature makes it an appropriate solution for many system and data integration projects. We consider a Service-oriented Architecture (SoA) implemented by Web Services to be a desireable approach for provisioning resources spanning from networks to devices.

This approach allows Eucalyptus to quickly build a toolbox that consists of tools developed by different vendors with different execution environments with a uniform interface. It is flexible and extensible: new resources can be simply added by creating and publishing new services or they can be removed by removing them from the resource registry.

Although Web Services can be used in both data integration and provisioning, there are a few differences. Provisioning Web Services mostly interact with the control flow of the applications, while Web Services for data integration mostly interact with the data flow, where data is wrapped in XML. Wrapping application data in XML is not appropriate in the broadband context, where the data streams are typically in the 1-1000 Mb range. In addition, the overhead of marking up data into XML-based SOAP message often hinders the number of Web Services in a service composition. On the other hand, provisioning parameters normally are very light-weight comparing to application data. Thus a provisioning Web Service workflow can comprise a number of nesting Web Services without affecting the efficiency of the system. Monitoring the status of applications and their underlying network is important to provide intelligent provisioning services such as comparing actual and desired states of an application, while this is not as critical in the data integration.

The Eucalyptus prototype is useful in assisting architects to do collaborative design in distributed labs [8] [9]. Although the current prototype is applied for architectural design, using Web Services in provisioning is indeed applicable to many industries. The value of Eucalyptus is also being recognized by users, tool vendors, system integrators and venture capitalists. CANARIE has already acted to establish Canadian leadership in the new application space of which Eucalyptus is the first example. St. Arnaud [15] believes that the agile infrastructure we created will have real potential and significant impact in many other fields and disciplines. As a result of our demonstrations, the Department of Public Safety in the Province of New Brunswick (Canada) is interested in having us apply our approach for responding to critical events; a few big companies are also interested in adopting the Eucalyptus platform for their industrial environments. In summary, this service-oriented approach can serve as a basic building block for an agile low-cost enterprise system without investing in expensive enterprise solutions. The service-oriented approach adopted by Eucalyptus makes provisioning, running, and monitoring heterogeneous networks and network-enabled resources relatively easy and intuitive.

Acknowledgements

This project is funded by CANARIE's Intelligent Infrastructure Program. Our partners include the Carleton Immersive Media Studio (CIMS) at Carleton University, Communication Research Centre Canada (CRC), IBM, Pleora Technologies Inc., and AutoDesk. This work is also contributed by Bo Xu and Libo Zhang, who are the previous development team members.

References

[1] Ibm deep computing visulization networking. http://www03.ibm.com/servers/deepcomputing/visualization/

[2] Arjen Poutsma, Rick Evans. Why Contract First? http://static.springframework.org/spring-ws/site/reference/html/why-contract-first.html, May 2007.

[3] Deborah McGuinness, Frank Harmelen. OWL Web Ontology Language Overview. http://www.w3.org/TR/owlfeatures/, February 2004.

[4] Don Box, David Ehnebuske, GopalKakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, May 2000.

[5] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Services Description Language 1.1. http://www.w3.org/TR/wsdl, March 2001.

[6] The Apache Software Foundation. Web service invocation framework. http://ws.apache.org/wsif/, May 2007.

[7] J. Wu et al. User-managed End-to-end Lightpath Provisioning over CA*Net4. In Proceeding of the National Fiber Optic Engineers Conference (NFOEC), Orlando, FL, USA, pages 275-282, September 2003.

[8] Michael Jemtrud, Martin Brooks, Bobby Ho, Sandy Liu, Philam Nguyen, John Spence, and Bruce Spencer. Eucalyptus: User controlled lightpath enabled participatory design studio. In ACADIA(The Association for Computer-Aided Design in Architecture)International Conference 2006, 10 2006.

[9] Michael Jemtrud, Philam Nguyen, Bruce Spencer, Martin Brooks, Sandy Liu, Yong Liang, Bo Xu, and Libo Zhang. Eucalyptus: Intelligent Infrastructure Enabled Participatory Design Studio. In WSC'06: Proceedings of the 37th conference on Winter simulation, pages 2047-2054. Winter Simulation Conference, 2006.

[10] Justin Murray. Learn the Eight Principles of Web Services Management. http://www.devx.com/enterprise/Article/10663/1954?pf=true, May 2007.

[11] Sean Bechhofer, Peter F. Patel-Schneider, Daniele Turi. OWL Web Ontology Language Concrete Abstract Syntax. http://owl.man.ac.uk/2003/concrete/latest/, 2003.

[12] Aaron Skonnard. Service station. http://msdn.microsoft.com/msdnmag/issues/05/05/ServiceStation/#S5, May 2007.

[13] SSEL. Web services management layer (wsml). http://ssel.vub.ac.be/wsml/, May 2007.

[14] Bill St.Arnaud. CA*net4 research program update - UCLP roadmap: Web Services workflow for connecting research instruments and sensors to networks. http://www.canarie.ca, December 2004.

[15] Bill St.Arnaud. Cyber-infrastructure and grids for Architecture Collaborative Design. http://lists.canarie.ca/pipermail/news/2006/000362.html, December 2006.

[16] Sun MicroSystems, Inc. Java Web Start Overview. http://java.sun.com/developer/technicalArticles/WebServices/JWS_2/JWS_White_Paper.pdf, May 2005.

[17] The CANARIE Inc. Live television - like never before! The CANARIE Times, December 2006. http://www.canarie.ca/times/Dec06/15.html.

[18] The OASIS Service Oriented Architecture TC. OASIS Reference Model for Service Oriented Architecture (Committee Draft)1.0. http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf, February 2007.

[19] The TIBCO Inc. Service mediation. http://www.tibco.com/resources/solutions/soa/esb_for_soa.pdf, May 2007.

[20] The UltraGrid Project team. UltraGrid: A High Definition Collaboratory. http://ltragrid.east.isi.edu/.

[21] James Thomson. DragThing. http://www.dragthing.com.